UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Detección de anomalías en el protocolo Modbus

TESIS
QUE PARA OPTAR POR EL GRADO DE:
Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:
Paulo Santiago de Jesús Contreras Flores

DIRECTOR DE TESIS:
Ing. Mario Rodríguez Manzanera
Instituto de Investigaciones en Matemáticas
Aplicadas y en Sistemas

CIUDAD UNIVERSITARIA, CD. DE MÉXICO. ENERO 2017

Resumen

Modbus es un protocolo de comunicación diseñado en la década de 1970, que es ampliamente utilizado en los Sistemas de Control Industrial (ICS) para la comunicación entre dispositivos de control de un proceso. Este protocolo no tiene forma de cifrar los datos, de revisar la integridad de los mensajes, o de establecer una autenticación entre los dispositivos que lo usan, es por ello que es necesario contar con mecanismos de seguridad que disminuyan los efectos que pudiera acarrear la explotación de las vulnerabilidades de estos sistemas. Por lo general las fallas en un ICS son consideradas graves debido a la criticidad de sus procesos industriales, al respecto se han registrado varias afectaciones a lo largo de la historia, por ejemplo en 1982 se registró un ciberataque al gasoducto Transiberiano, o el tan conocido gusano Stuxnet que afectó una planta de enriquecimiento de uranio en Irán en el año 2009. Por otra parte, la detección de anomalías se refiere al modelo de detección de intrusos que consiste en el análisis del tráfico de red para reportar comportamiento anómalo o inusual respecto a los patrones de tráfico normal para un sistema. Esta investigación se centró en identificar las conexiones anómalas, causadas por un ciberataque de suplantación de uno de los dispositivos del entorno de pruebas de ICS, de las conexiones del comportamiento normal en dicho entorno. Se usó como método de detección de anomalías al clasificador ingenuo de Bayes aplicándolo al tráfico de red anómalo y normal, este tráfico se obtuvo de la interconexión de los dispositivos del sistema de pruebas.

Agradecimientos

Gloria Patri, et Filio, et Spiritui Sancto. Sicut erat in principio, et nunc, et semper, et in saecula saeculorum. Amen

A mi mamá Rosario y mi papá Fortino, a mi hermano Felipe, a mi hermano Fortino y su familia, Aurora y Angelito, por su cariño, paciencia, comprensión y apoyo que siempre me han brindado. A mis tías Tere y Pana porque siempre nos han acompañado y apoyado a mí y a mi familia.

A mis amigos con los que siempre he podido contar, por pasar buenos momentos conmigo y darme su amistad. Particularmente a todos aquellos que de alguna manera me apoyaron para la realización de este trabajo.

A mis sinodales, Dra. María Elena Lárraga Ramírez, M. en C. José Roberto Sánchez Soledad, Dr. Christopher Rhodes Stephens Stevens, Dr. Javier Gómez Castellanos y al Ing. Mario Rodríguez Manzanera, por el tiempo y apoyo que me dieron para la conclusión de esta investigación.

A la Universidad Nacional Autónoma de México, a la Facultad de Ingeniería, al Posgrado en Ciencia e Ingeniería en Computación y al UNAM-CERT por formarme tanto en el ámbito profesional como también en el humano.

Reconozco al Consejo Nacional de Ciencia y Tecnología por el financiamiento económico brindado para la realización de mis estudios de posgrado, dentro de su obligación conferida de impulsar las actividades científicas y tecnológicas del país.

Índice general

Índice de figuras **9**

Índice de tablas **11**

1. Introducción **13**
1.1. Estado del arte . . . 13
1.2. Hipótesis de investigación . . . 16
1.2.1. Hipótesis específicas . . . 17
1.3. Objetivos . . . 17

2. Sistemas de Control Industrial **19**
2.1. Resumen . . . 19
2.2. ¿Qué son los Sistemas de Control Industrial (ICS)? . . . 19
2.2.1. Principales dispositivos de ICS . . . 20
2.2.2. Protocolos Modbus y DNP3 para ICS . . . 22
2.3. Problemas de seguridad en ICS . . . 24

3. Protocolo Modbus TCP/IP **29**
3.1. Resumen . . . 29
3.2. Origen del protocolo Modbus . . . 29
3.3. El Modelo TCP/IP . . . 30
3.4. Modelo cliente-servidor y Modbus TCP/IP . . . 31
3.5. Paquetes Modbus TCP/IP . . . 32
3.6. Modelo de datos de Modbus . . . 35
3.7. Funciones de Modbus . . . 36
3.8. Ejemplo de una comunicación de Modbus TCP/IP . . . 38
3.8.1. Análisis del Three-way handshake . . . 38
3.8.2. Análisis de Request-Response . . . 40
3.9. Recomendaciones de implementación para Modbus TCP/IP . . . 43

4. Seguridad en el protocolo Modbus TCP/IP **47**
4.1. Resumen . . . 47
4.2. Seguridad de la información. . . . 47
4.3. Problemas de seguridad en el protocolo Modbus . . . 48
4.4. Taxonomía de ataques al protocolo Modbus . . . 50
4.4.1. Ataques a Modbus Serial y TCP/IP . . . 51
4.4.2. Ataques a Modbus TCP/IP . . . 51

5. Técnicas de detección de intrusos **53**
5.1. Resumen . . . 53
5.2. Basadas en firmas . . . 53
5.3. Detección de anomalías . . . 54
5.3.1. Clasificador ingenuo de Bayes . . . 55
5.4. Métricas de evaluación . . . 58
5.4.1. Curvas ROC . . . 60

6. Obtención de datos **63**
6.1. Resumen . . . 63
6.2. Fuentes de datos de ICS . . . 63
6.3. Infraestructura de ICS para pruebas . . . 67

7. Análisis de los datos **71**
7.1. Resumen . . . 71
7.2. Procesamiento de los datos . . . 71
7.3. Análisis . . . 74

8. Conclusiones **79**
8.1. Trabajo Futuro . . . 81

Bibliografía **83**

A. Glosario **87**

B. Interpretación de los datos de los PLC **89**
B.1. PLC para un fotosensor . . . 89
B.1.1. Función *Write Multiple Coils*. . . . 89
B.1.2. Función *Read Holding Registers*. . . . 91
B.2. PLC para un semáforo . . . 94
B.2.1. Función *Write Multiple Coils*. . . . 94

C. Circuitos **97**

D. Diccionario de datos **99**
D.1. Variables utilizadas . . . 99
D.2. Variables descartadas. . . . 101
D.3. Valores utilizados . . . 102

Índice de figuras

1-1. Arquitectura de un IDS [García-Teodoro et al., 2009] 14

2-1. Funcionamiento e integración del HMI. [modificada de Knapp and Langill, 2015] 21
2-2. Estructura general de un ICS. [McLaughlin et al., 2016] 22
2-3. Vulnerabilidades descubiertas en productos específicos por año. [FireEye, 2016] 26
2-4. Modelo simplificado de Purdue de un ICS. [FireEye, 2016] 27
2-5. Vulnerabilidades por Zonas, descubiertas entre febrero de 2013 y abril de 2016. [FireEye, 2016] 27

3-1. Correspondencia entre los modelos OSI y TCP/IP. 30
3-2. Comunicación entre dos dispositivos a través del protocolo Modbus, usando el modelo TCP/IP. [modificada de Fall and Stevens, 2012] 31
3-3. Modelo cliente-servidor para Modbus 32
3-4. Modbus TCP/IP Application Data Unit 33
3-5. Arquitectura Modbus para TCP/IP [Jiménez Díaz, 2011] 33
3-6. Transacción Modbus sin error [Organization, 2012] 34
3-7. Transacción Modbus con una excepción como respuesta [Organization, 2012] . 34
3-8. Three-way handshake 39
3-9. Datos en Modbus PDU para el PLC del fotosensor 40

5-1. Firma del IDS Snort 53
5-2. Matriz de confusión 58
5-3. Curva ROC 60

6-1. Diagrama lógico de la infraestructura de la captura de tráfico de CloudShark. 65
6-2. Diagrama lógico de la infraestructura de la captura de tráfico usando el Compact RIO 9074. 65
6-3. Infraestructura para el Compact RIO 9074. 66
6-4. Infraestrucura ICS para 4SICS. 66
6-5. Diagrama lógico de la infraestructura de ICS de pruebas. 67
6-6. Infraestructura de pruebas. 68

7-1. Bytes totales 72
7-2. Curva ROC para el clasificador 77
7-3. Distribución del tráfico anómalo y normal 77

B-1. Datos en Modbus PDU para el PLC del fotosensor. Función *Write Multiple Coils*. 89

B-2. Datos en Modbus PDU para el PLC del fotosensor. Función *Read Holding Registers* 92
B-3. Significado de los datos en Modbus PDU para el PLC del semaforo. Función *Write Multiple Coils*. 94

C-1. Diagrama de conexión del circuito fotosensor 97
C-2. Esquema eléctrico de conexión del circuito fotosensor 97
C-3. Diagrama de conexión del circuito semáforo 98
C-4. Esquema eléctrico de conexión del circuito semáforo 98

Índice de tablas

3.1. Tipos de PDU en Modbus. 35
3.2. Modelo de bloques de almacenamiento de datos en Modbus [Organization, 2012] 35
3.3. Extracto de los códigos de función públicos. 36
3.4. PDU del *Request* para el código de función 3. 37
3.5. PDU del *Response* para el código de función 3. 37
3.6. PDU de *Excepción* para el código de función 3. 37
3.7. PDU del *Request* para el código de función 15. 37
3.8. PDU del *Response* para el código de función 15. 38
3.9. PDU de *Excepción* para el código de función 15. 38

7.1. Características de la captura de tráfico anómalo 72
7.2. Características de la captura de tráfico normal 72
7.3. Archivos de entrenamiento y prueba 73
7.4. Ejemplos de variables con *Score* positivo 74
7.5. Ejemplos de variables con *Score* cercano al cero 75
7.6. Ejemplos de variables con *Score* negativo 75
7.7. Matriz de confusión para los flujos de pruebas 76
7.8. Resultados de las métricas de evaluación 76

D.1. Variables utilizadas en el análisis (1) 99
D.2. Variables utilizadas en el análisis (2) 100
D.3. Variables descartadas para el análisis 101
D.4. Valores de las variables utilizadas (1) 102
D.5. Valores de las variables utilizadas (2) 103
D.6. Valores de las variables utilizadas (3) 104

Capítulo 1

Introducción

El estudio de la seguridad de la información ha cobrado mayor interés en la última década debido a que se ha demostrado que su ámbito se ha extendido cada vez a más áreas críticas del quehacer humano. Se han reportado con evidencia sólida ataques a dispositivos de control automatizado como los empleados en la industria del petróleo y gas, de la energía nuclear, de control de tránsito, y química entre otros. Muchos de estos sistemas industriales utilizan protocolos de comunicación diseñados hace décadas por lo que no se consideraron aspectos desde el punto de vista de la seguridad de la información. Por ejemplo, el protocolo Modbus es ampliamente utilizado en dichos sistemas industriales para la comunicación entre dispositivos de control sencillos y equipos complejos, su función puede ser la de enviar datos de mediciones de sensores, enviar datos con instrucciones de control, entre otras funciones, no fue diseñado con medidas de seguridad por lo que es necesario que se lleven a cabo en las redes de control industrial políticas y procedimientos que permitan llevar a cabo la tarea de la seguridad de la información. Por eso se requieren de técnicas para detectar tráfico de Modbus malicioso, ésta es la principal motivación de este trabajo de investigación.

Por otro lado, se ha extendido el uso de algoritmos y métodos para el análisis de datos complejos de diversas áreas con el fin de llevar a cabo análisis que permitan entender las causas de fenómenos o incluso de poder predecirlos. Para llevar a cabo estos estudios se usan áreas de cómputo como Aprendizaje automatizado, Minería de datos, Reconocimiento de Patrones, por mencionar las principales. Estas técnicas también pueden ser aplicadas al estudio de los patrones de las acciones que derivan en una brecha en la seguridad de un Sistema de Control Industrial.

1.1. Estado del arte

La detección de intrusos en redes de computadoras se refiere a los métodos empleados para encontrar comportamientos maliciosos en el tráfico de red de un sistema. En la literatura relacionada se distinguen dos principales métodos de detección de intrusos de acuerdo al tipo de análisis llevado a cabo, basadas en firmas o basadas en anomalías. Estos métodos se implementan en un Sistema de Detección de Intrusos (*Intrusion Detection System*, IDS) que es el encargado de monitorear los eventos que ocurren en una computadora o en una red de computadoras, y analizarlos en búsqueda de actividad maliciosa [Bhattacharyya and Kalita, 2014]. De acuerdo a [García-Teodoro et al., 2009] un Sistema de Detección de Intrusos está conformado por cuatro módulos funcionales que se muestran en la figura 1-1, y estos son

- Eventos (*Event boxes*). Se compone de los sensores que monitorean al sistema, estos sensores adquieren datos correspondientes a los eventos que serán analizados por otros módulos.
- Base de datos (*Databases boxes*). Este módulo almacena información proveniente del módulo de Eventos, dicha información será procesada por los módulos de Análisis y de Respuesta.
- Análisis (*Analysis boxes*). Módulo que analiza los eventos con el fin de detectar posible comportamiento malicioso.
- Respuesta (*Response boxes*). Si ocurre una intrusión en el sistema este módulo es el encargado de dar una respuesta ante la amenaza detectada.

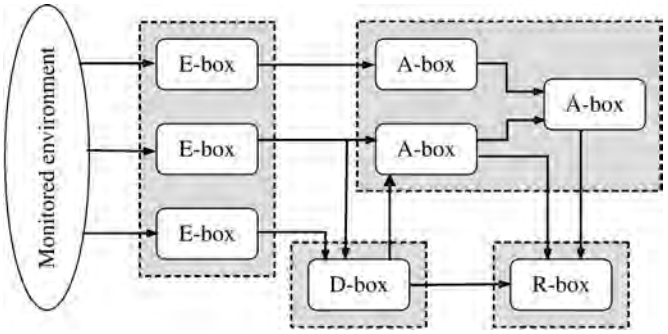


Figura 1-1: Arquitectura de un IDS [García-Teodoro et al., 2009]

Si los datos que se reciben en el Módulo de eventos corresponden a identificadores de procesos, llamadas al sistema o cualquier otra información referente al sistema operativo de una computadora entonces se dice que es un IDS de *host* o HIDS, en cambio si los datos corresponden a cantidad de tráfico, direcciones IP, puertos, protocolos, y demás información de tráfico de red, se dice que es un IDS de red o NIDS, en este trabajo solamente se mencionarán a los IDS de red. Los métodos de detección basados en firmas y basados en anomalías se ejecutan en el Módulo de análisis.

Así pues, los métodos de detección de intrusos basados en firmas consisten en la creación de patrones específicos de tráfico anómalo o malicioso previamente conocido, la detección se realiza cuando el tráfico analizado coincide con el patrón creado. Este método falla ante nuevos ataques porque el NIDS no cuenta aún con firmas que detecten estos nuevos patrones. En cuanto a la implementación de este método de detección, existen en el mercado tanto software libre como el desarrollado de forma propietaria por empresas, como ejemplo de NIDS de software libre se tiene a Snort, Suricata o Bro Network Security Monitor, como ejemplo de NIDS propietarios se tiene a las marcas SourceFire (basado en Snort), Hewlett Packard, Palo Alto Network, TippingPoint, entre otros. Han sido varios los esfuerzos de la comunidad de software libre de crear firmas eficaces y de fácil acceso para detectar tráfico malicioso de Sistemas de Control Industrial usando un NIDS, a continuación se relatan tres ejemplos.

El primer ejemplo se refiere al trabajo desarrollado por [Jiménez Díaz, 2011] en donde explica a fondo cómo elaborar firmas para el NIDS Snort, se deben de tomar en cuenta

direcciones IP, puertos, protocolos, además de las funciones y datos propios del protocolo Modbus para crear las firmas. Por otro lado, la intención de esta publicación también es la de difundir una metodología para que los encargados de la seguridad de compañías con este tipo de sistemas puedan contar con una protección sin tener que elevar los costos, ya que el gasto en productos comerciales puede no ser el adecuado para todas las compañías, asimismo afirma que algunos productos comerciales no son flexibles ni claros para el usuario.

Como segundo ejemplo se tiene una serie de firmas también para el NIDS Snort elaboradas bajo el proyecto QuickDraw [Digital Bond, 2011], con estas firmas se busca detectar tráfico malicioso de los protocolos Modbus, DNP3, y EtherNet/IP, y también tráfico de la posible explotación de vulnerabilidades descubiertas en dispositivos específicos de Sistemas de Control Industrial. Las recomendaciones dadas es que se usen solamente las firmas que se ajusten al tipo de Sistema de Control Industrial de cada empresa. Estas firmas permiten la identificación de solicitudes no autorizadas, solicitudes y respuestas malformadas de los protocolos, comandos peligrosos y no usuales, y de otras acciones potencialmente maliciosas.

Por último, en el tercer ejemplo se detalla sin ser específicos en algún tipo de tecnología lo cual permite implementarlas en cualquier NIDS, cómo elaborar cincuenta firmas obtenidas a partir del análisis de vulnerabilidades del protocolo Modbus, son descritas en [Morris et al., 2013]. Las firmas están diseñadas para implementarse tanto en redes que usen Modbus TCP como en redes que usen Modbus Serial.

Con relación al modelo de detección de intrusos basado en anomalías, éste consiste en el análisis del tráfico de red para reportar comportamiento anómalo o inusual respecto a los patrones de tráfico normal para un sistema. Existen tres principales técnicas de detección de anomalías de acuerdo al comportamiento del sistema a analizar, basado en estadísticas, base de conocimiento y aprendizaje automatizado [García-Teodoro et al., 2009]. El propósito del método de detección de anomalías es analizar, entender y caracterizar el comportamiento del tráfico de red, para clasificar o identificar los registros de tráfico anómalo o anormal de los registros normales; desde el punto de vista del Aprendizaje automatizado, el problema de la detección de anomalías es un problema de clasificación [Bhattacharyya and Kalita, 2014]. También para este método de detección se han desarrollado trabajos académicos, a continuación se hace una revisión de algunos de ellos.

En la investigación de [Bhatia et al., 2014] se utilizó como método de detección de anomalías una variante del algoritmo EWMA (*Exponentially Weighted Moving Average*) para detectar ataques de inundación de paquetes a una infraestructura de pruebas de Sistemas de Control Industrial, este algoritmo puede detectar cambios abruptos en los parámetros del tráfico de red. En el artículo se explica que se escogió este algoritmo porque es simple, flexible, robusto y efectivo, especialmente para detectar ataques de alta intensidad como un ataque de inundación, además de tener una tasa baja de falsos positivos cuando se analizan grandes cantidades de datos, en comparación con el algoritmo CUSUM (*Cumulutive Sum*). El proceso industrial a monitorear está conformado por dos tanques para agua con medidores de presión y temperatura, con conexiones entre ellos para transportar agua de un tanque a otro y viceversa, el flujo de agua está controlado por válvulas y una bomba. Se utiliza el Compact RIO 9074 de la marca National Instruments como PLC y funge como esclavo, como maestro que recibirá y enviará datos de control al esclavo se utiliza una computadora con el software LabView, el protocolo para comunicar a ambos es Modbus TCP. El ataque

de inundación de paquetes se realiza con un programa que envía solicitudes al esclavo para determinar qué tipo de registros de Modbus se están utilizando, después el atacante puede enviar solicitudes para intentar controlar el sistema, este programa permite crear además una inundación de paquetes de solicitudes dirigidas al PLC. Cuando se ejecutó el ataque a la infraestructura, se logró apagar y encender la bomba de agua a voluntad al inundar de paquetes al PLC. La variante del algoritmo EWMA fue capaz de detectar los ataques realizados, aunque una vez que se presentaron los ataques tardó en detectarlos, sin embargo fue capaz de adaptarse a los cambios en la cantidad de paquetes del tráfico de red, detectando cuando comenzaba y cuando se detenía el ataque.

Siguiendo con la revisión de investigaciones sobre detección de anomalías en el protocolo Modbus, se encuentra el trabajo de [Goldenberg and Wool, 2013] en donde para caracterizar el comportamiento de un Sistema de Control Industrial crean una Autómata Finito Determinista por cada relación esclavo-maestro, cuyos estados y función de transición se obtiene de la observación del comportamiento de cerca de 100 mensajes capturados. Este modelo de detección es altamente sensitivo al identificar como anomalía cualquier comportamiento que se salga de la secuencia normal del autómata. Una de las ventajas de este trabajo es que los datos fueron obtenidos de un ambiente real de Control Industrial, el tráfico de red del protocolo Modbus se obtuvo de la Planta de energía de la Universidad de Tel Aviv en Israel. Las anomalías presentadas fueron causadas por el trabajo de técnicos cuando solucionaban problemas en el Sistema. Como resultado se obtuvo una tasa baja de falsos positivos a pesar de ser un modelo altamente sensitivo, del tráfico de los siete PLC monitoreados, se obtuvo una detección del 100 % en cinco de ellos, además permite detectar anomalías reales que no son maliciosas; por otro lado será necesario probar este modelo con ataques que busquen interferir con el comportamiento del Sistema, y también probarlo en otros Sistemas de Control Industrial que operen con el protocolo Modbus TCP para conocer el comportamiento de detección.

Finalmente, el estudio realizado por [Santillan, 2014] se basa en establecer Árboles de patrones de las comunicaciones del Sistema de Control Industrial de interés, usando secuencias cortas y un valor de ventana deslizante. Dichas secuencias están conformadas por el Identificador de unidad, Código de función y Datos, obtenidos de la cabecera del protocolo Modbus. Se modificó el algoritmo original para poder adaptarlo al contexto de Sistemas de Control Industrial. Los resultados de detección son considerados aceptables para los datos de entrenamiento, al no presentarse falsos positivos. Se deja abierto el trabajo para mejorarlo en un futuro considerando el análisis del comportamiento para valores diferentes de ventana en datos no entrenados.

1.2. Hipótesis de investigación

Aplicando un método de detección de anomalías al tráfico de red de un entorno de pruebas de Sistemas de Control Industrial se podrá identificar las conexiones con comportamiento anómalo de las conexiones con comportamiento normal.

1.2.1. Hipótesis específicas

- Se obtendrá tráfico de red de un entorno de pruebas de Sistemas de Control Industrial que use como protocolo de comunicación a Modbus, este tráfico de red será tanto de comportamiento normal como de comportamiento anómalo del sistema. Para obtener el comportamiento anómalo se ejecutará un ataque informático al entorno de pruebas.
- Aplicando el clasificador ingenuo de Bayes a los flujos de red del tráfico obtenido del entorno de pruebas, se podrá identificar las conexiones anómalas de las conexiones normales de acuerdo a los patrones de comportamiento.

1.3. Objetivos

- Obtener un entorno de pruebas de Sistemas de Control Industrial que utilice a Modbus como protocolo de comunicación, de bajo costo, escalable, de hardware y software abierto.
- Aplicando el clasificador ingenuo de Bayes, obtener las variables de mayor influencia a través de su valor de *Score*, que contribuyan a determinar si un flujo del tráfico de red previamente capturado es anómalo o no.
- Con base en métricas de evaluación, determinar si es viable utilizar el clasificador ingenuo de Bayes como método de detección de anomalías en un NIDS para el entorno de pruebas.

En este trabajo de investigación primero se revisa en el Capítulo 2 lo que es un Sistema de Control Industrial, sus principales componentes, las brechas de seguridad que se han reportado y la importancia de su estudio. En los dos capítulos siguientes se estudia a detalle al protocolo de comunicación Modbus el cual es utilizado ampliamente en los sistemas industriales, en el Capítulo 3 se estudia su especificación y se explica a través de un ejemplo práctico sus principales características, mientras que en el Capítulo 4 se estudian sus vulnerabilidades y una taxonomía de ataques para este protocolo. Las técnicas de detección de intrusos se tratan en el Capítulo 5, se da un vistazo general a las técnicas clásicas de detección basadas en la formulación de patrones de tráfico conocidas como firmas, para proseguir con las técnicas de detección basadas en anomalías, especialmente la que se basa en el uso del clasificador ingenuo de Bayes. Se dedica todo el Capítulo 6 a las dificultades presentadas para obtener los datos de Sistemas de Control Industrial, las pruebas realizadas y la solución final. En el Capítulo 7 se presenta la forma en cómo se analizaron los datos de Control Industrial y se discuten los resultados. Por último en el Capítulo 8 se abordan las conclusiones. Se presentan también cuatro apéndices, el A con el glosario, el B con una explicación detallada de ejemplos de transacciones Modbus, el C con las especificaciones para construir los PLC basados en Arduino y los circuitos que éstos controlan, y por último el D en donde se detallan las variables y sus valores asociados, utilizadas en el clasificador ingenuo de Bayes.

Se pone a disposición de cualquier persona interesada los programas utilizados para crear la infraestructura de pruebas presentada, el programa para enviar paquetes anómalos y los datos analizados, en el sitio `https://github.com/nehnemini/mb_anomaly_detection`.

Capítulo 2

Sistemas de Control Industrial

2.1. Resumen

El objetivo de este capítulo es definir qué son los Sistemas de Control Industrial, cuales son sus principales componentes y cuales son los protocolos de comunicación más utilizados por la industria, dar un panorama general de los problemas de seguridad relacionados con este tipo de sistemas y el impacto que provocan, de esto se establece la importancia del análisis presentado a lo largo de esta tesis. Se han estudiado diferentes fuentes para presentar la información, principalmente se ha utilizado a [Knapp and Langill, 2015] porque presenta de un forma amplia y detallada a los Sistemas de Control Industrial, además se ha consultado diversos artículos de revistas indexadas y reportes de compañías y organizaciones especializadas en el área de la seguridad de la información.

2.2. ¿Qué son los Sistemas de Control Industrial (ICS)?

Un Sistema de Control Industrial (ICS por las sigles en inglés de Industrial Control System) es un sistema que combina componentes eléctricos, mecánicos, hidráulicos, etc, para el control de procesos industriales. Los ICS son utilizados principalmente en la industria eléctrica, sistemas de agua y de drenaje, de gas natural, petrolera, química, de transporte, farmacéutica, de celulosa y papel, alimentos y bebidas, automovilística, aeroespacial y energética. A la parte principal del sistema, es decir, al objetivo específico de una industria, se le conoce como el proceso industrial. [Stouffer et al., 2015]

Un ICS está conformado por varios tipos de sistemas, estos pueden ser Sistema de Control del Proceso (Process Control System, PCS), Sistema Distribuido de Control (Distributed Control Systems, DCS), Sistema de Control de Supervisión y Adquisición de Datos (Supervisory Control and Data Acquisition, SCADA), y el Sistema de Protección Instrumentado (Safety Instrumented Systems, SIS), [Knapp and Langill, 2015].

En un Sistema Distribuido de Control o DCS, los dispositivos de control automatizado se distribuyen a lo largo del sistema, estos dispositivos están conectados a una red para monitorear y supervisar el proceso a distancia, es decir, un DCS se despliega y controla de forma distribuida. Cada proceso de control distribuido se controla individualmente, de tal forma que si una parte del sistema de control falla, las demás partes del DCS pueden seguir funcionando, [Knapp and Langill, 2015] y [McLaughlin et al., 2016].

Un Sistema de Control de Supervisión y Adquisición de Datos o SCADA, es un sistema compuesto por dispositivos de cómputo y redes usado para monitorear y controlar procesos industriales. A través de las redes se comunica con los dispositivos de control ubicados remotamente, para adquirir datos de éstos y proporcionarlos a los operadores con el fin de supervisar el proceso industrial. Una vez que se tienen estos datos se toman acciones de supervisión para en su caso, realizar ajustes en los dispositivos de control. Este monitoreo y control se tiene que llevar a cabo en tiempo real, [Knapp and Langill, 2015] y [McLaughlin et al., 2016].

2.2.1. Principales dispositivos de ICS

Los Sistemas de Control Industrial están conformados por varios dispositivos interconectados entre sí para constituir todo el sistema, a continuación se describen los principales.

- *Programmable Logic Controller* (PLC). Los Controladores Lógicos Programables o PLC, son dispositivos de cómputo de propósito especifico cuya función es automatizar procesos en tiempo real de tipo industrial, por ejemplo el control automático de semáforos, en la industria química, petrolera, eléctrica, de gas. A diferencia de las computadoras, los PLC no utilizan un sistema operativo común, sino programas específicos los cuales permiten que ante entradas, producidas por ejemplo por un sensor de presión, se generen salidas específicas, por ejemplo abrir una válvula de un tanque de un fluido para disminuir o aumentar la presión, es decir, usa valores de entrada y salida junto con lógica programable para llevar a cabo el control automático de sensores, actuadores, y dispositivos mecánicos como válvulas, etc. Fueron diseñados originalmente para reemplazar a componentes electromecánicos. Los PLC pueden usar distintos protocolos para comunicarse con los dispositivos a los cuales controla al enviar y recibir datos, los protocolos más comunes para llevar a cabo esto son Modbus, ControlNet, EtherNet/IP, PROFIBUS y PROFINET. [Knapp and Langill, 2015]

- *Programmable Automation Controllers* (PAC). Los Controladores de Automatización Programable o PAC, se pueden considerar como PLC más completos, es decir, con más funciones. Se conforma por un controlador, comúnmente una CPU, módulos de entradas y salidas, y uno o múltiples conectores de datos. Combina las características de control automático de un PLC junto con las funciones de monitoreo, cálculo y desempeño de una computadora de tipo industrial. [Beckhoff Sist. Industriales, 2016]

- *Remote Terminal Unit* (RTU). Una Unidad Terminal Remota o RTU, monitorea parámetros de dispositivos que se encuentran ubicados remotamente, y los envía a una estación central de monitoreo, ya sea a una Unidad Terminal Maestra o MTU, que puede ser un servidor ICS, a un PLC central, o directamente a un HMI. Las RTU incluyen por lo general la posibilidad de comunicación remota a través de un módem, conexión celular, de radio, u otra tecnología de comunicación remota. Se le puede considerar a un RTU como un PLC que tiene integradas funciones de comunicación remota. Este tipo de dispositivos es común encontrarlos en las industrias en las que se usan fluidos como líquidos o gases. [Knapp and Langill, 2015]

- *Intelligent Electronic Device* (IED). Un Dispositivo Electrónico Inteligente es utilizado en la industria eléctrica, en donde es utilizado en áreas con alto voltaje, en donde se generan señales de interferencia o ruido eléctrico propio de estos ambientes. Su función es similar a la de los PLC y RTU, sólo que está diseñado específicamente para la industria eléctrica. [Knapp and Langill, 2015]

- *Human-Machine Interface* (HMI). Una Interfaz Humano-Máquina o HMI, es un dispositivo que es utilizado como un medio para interactuar con los PLC, PAC, RTU e IED al reemplazar a controles eléctricos como switch, perillas, etc, con representaciones gráficas de controles digitales utilizados para sensar y manipular el proceso de control industrial. Debido a que los HMI están conformados por una parte de software, se reemplaza a cables y controles físicos con parámetros de software, permitiendo que se adapten de forma fácil y sencilla a diferentes tipos de industrias. Actúa como un puente entre el operador, que es una persona, y la lógica compleja de uno o más PLC. Generalmente el HMI cuenta con una interfaz gráfica que representa al proceso que se está controlando, incluyendo valores de sensores y otras medidas, y representación de los diferentes estados de las salidas [Knapp and Langill, 2015]. En la figura 2-1 se muestra la integración del HMI con los controladores y los dispositivos físicos de tipo industrial.

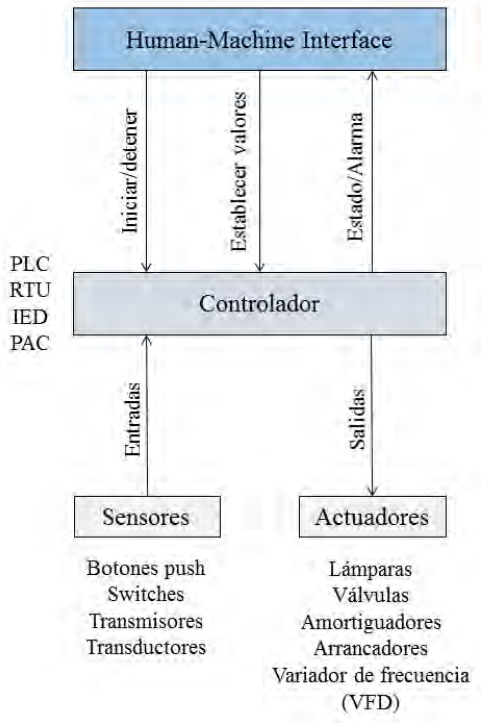


Figura 2-1: Funcionamiento e integración del HMI.
[modificada de Knapp and Langill, 2015]

- Estación de supervisión o *Workstation*. Recopilan información de los dispositivos utilizados en los ICS, o de la base de datos históricos del ICS. Son principalmente de sólo lectura, [Knapp and Langill, 2015].

- Almacenamiento histórico de datos o *Data historian*. Es un sistema que recolecta y almacena datos provenientes de dispositivos de ICS. Por lo general cada fabricante de dispositivos de ICS cuenta con su propia base de datos, sin embargo hay fabricantes que ofrecen sistemas de almacenamiento de datos genéricos los cuales se pueden configurar para recibir datos de sistemas de otros fabricantes, [Knapp and Langill, 2015].

En la figura 2-2 se muestra un diagrama de cómo está conformada de forma general, una estructura de ICS. El proceso industrial está conectado a los dispositivos remotos (*Field devices*) como son los PLC, RTU, IED, éstos envían y reciben información a través de diferentes medios de comunicación ya sean cableados o inalámbricos (*Wired Wireless communication links*), hasta el centro de control (*Control Center*), en donde se lleva a cabo el análisis y almacenamiento y/o control de los datos del proceso, por lo tanto el control del proceso industrial mismo. Esta comunicación se lleva a cabo usando protocolos de comunicación estándar o propietarios como por ejemplo Modbus TCP/IP, DNP3, entre otros.

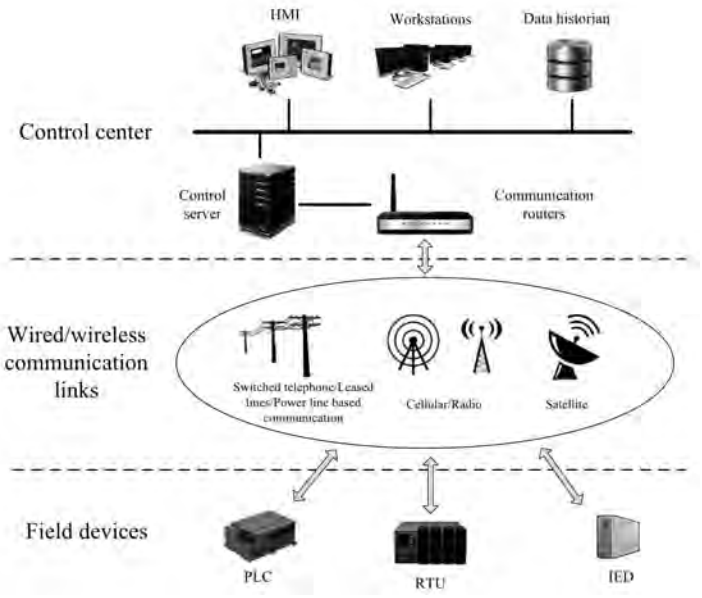


Figura 2-2: Estructura general de un ICS. [McLaughlin et al., 2016]

Los dispositivos aquí mencionados son los más comunes para los ICS, sin embargo existen otros dispositivos que también forman parte de un ICS.

2.2.2. Protocolos Modbus y DNP3 para ICS

Los dispositivos de ICS requieren de protocolos especializados para establecer comunicación entre ellos, estos protocolos han sido diseñados para ser eficientes y confiables por la criticidad de la mayoría de los procesos industriales en los que se usan además, es necesario que estos protocolos respondan en tiempo real para poder llevar a cabo el proceso industrial con la precisión y capacidad de respuesta necesarias. Existen diversos protocolos de comunicación de ICS, varios de ellos son estándares en la industria, otros han sido adoptados como estándares sin que en un inicio hayan sido pensados para serlo, y algunos otros han sido creados expresamente para funcionar en dispositivos de cierto fabricante y con un propósito específico.

De acuerdo a [McLaughlin et al., 2016] los protocolos de ICS se pueden dividir en dos categorías para su estudio, la primera se refiere a aquellos protocolos que se usan en el proceso industrial para el control de los dispositivos, por ejemplo se usan para conectar dispositivos que interactúan directamente con el proceso industrial como sensores, hacia dispositivos de control como los PLC, o a dispositivos de supervisión de los sistemas como HMI, Servidores ICS, etc; la segunda categoría se refiere a aquellos protocolos que permiten la comunicación entre sistemas como por ejemplo, la comunicación entre la base de datos de los datos históricos, hacia un servidor de ICS, o para conectar dos centros de control de ICS. En este trabajo solamente se estudiará a detalle el protocolo Modbus el cual pertenece a la primera categoría. A continuación se explica de forma general dos de los protocolos más utilizados en la industria, Modbus y DNP3, ambos pertenecen al tipo de protocolos de la primera categoría.

- *MODICON Communication Bus* o Modbus. El protocolo Modbus fue diseñado en 1979 para la comunicación en tiempo real entre dispositivos que controlan un proceso industrial. Modbus ha sido adoptado por la industria como un estándar *de facto*, y ha tenido mejoras a lo largo de los años. Esta adopción por la industria se debe a que parte de las características de Modbus permite el envío de mensajes a nivel de bits, no cuenta con restricciones de autenticación, requiere poco recursos de cómputo para su procesamiento, es un estándar abierto, se distribuye libremente, y recibe soporte por parte de la Organización Modbus [1]. Dispositivos sencillos utilizan a Modbus para comunicarse con equipos más complejos, que pueden leer las mediciones, realizar tareas de control y análisis de los datos, por ejemplo se usa para comunicar a los PLC o RTU con un sistema de supervisión de ICS, [McLaughlin et al., 2016]. Este trabajo se dedica al estudio de la detección de anomalías en este protocolo, el capítulo 3 trata con detalle al protocolo Modbus.

- *Distributed Network Protocolo* o DNP3. Este protocolo fue diseñado por la compañía Westronic en 1990, el propósito principal del diseño de este protocolo fue para proporcionar comunicaciones confiables en la industria eléctrica, en donde se presentan alto niveles de interferencia electromagnética y se contaban con medios de transmisión de datos poco confiables, como las líneas telefónicas análogas. DNP3 fue modificado para trabajar con los protocolos TCP y UDP, a través de paquetes IP en 1998, esto permitió que se extendiera su uso a industrias del petróleo y gas, de agua y drenaje. Algunas de las razones por las que la industria migró del protocolo Modbus a DNP3 fue que éste incluye el envío de informes de excepción, indicadores de calidad de datos, datos con marca de tiempo permitiendo llevar una secuencia de los acontecimientos, cuenta con controles de redundancia cíclica (Cyclical Redundancy Check, CRC) para verificar la integridad de los datos, incluye la opción de generar mensajes de confirmación para la capa de enlace, con lo que se incrementa la confiabilidad en el protocolo, en algunas variaciones del protocolo se incluye también soporte para autenticación a nivel de la capa de enlace. DNP3 es muy confiable, sin dejar de ser eficaz y adecuado para la transferencia de datos en tiempo real esto gracias a que utiliza varios formatos estándares para los datos; permite una comunicación bidireccional entre dispositivos es decir, soporta comunicaciones tanto del maestro hacia el esclavo, como del esclavo

[1] La Organización Modbus es un grupo de usuarios independientes y proveedores de dispositivos de automatización que impulsa el uso del protocolo de comunicación Modbus, así como también provee de información, normas de aplicación y certificación sobre el protocolo.

hacia el maestro, y permite el envío de informes basados en excepciones, por ejemplo, un dispositivo puede informar al centro de control de un evento fuera del intervalo normal aún cuando esta notificación no le haya sido solicitada, considerando esto como una condición de alarma. Los datos de los dispositivos del ICS que se envían en los mensajes de DNP3 se pueden utilizar para transferir información solamente de lectura, para enviar funciones de control, o para enviar datos binarios o analógicos directamente hacia los dispositivos como RTU o IED. [McLaughlin et al., 2016]

2.3. Problemas de seguridad en ICS

El proceso industrial en un Sistema de Control Industrial puede verse afectado intencionalmente por un ciberataque provocando la interrupción, falla o modificación de dicho proceso. Los protocolos industriales, deben de contar con características que permitan que la operación del proceso industrial se llevo a cabo de forma confiable y eficiente, debido a la criticidad del proceso, es por eso que la mayoría de estos protocolos no toman en cuenta como característica primordial la seguridad, ausentando características básicas como la autenticación o el cifrado; el incluir estas características de seguridad añade mayor procesamiento de cómputo para los dispositivos de ICS, siendo una razón más para que se deje de lado este aspecto. [McLaughlin et al., 2016]

Los protocolos usados en los ICS no fueron diseñados en un principio para operar en redes que fueran accesibles fácilmente por otros dispositivos que no fueran de ICS, por ejemplo la conexión entre dispositivos de control y otros dispositivos como impresoras, PC, etc, en una misma red, usando como protocolos a la pila TCP/IP, es por eso que la seguridad no fue tomada en cuenta como un papel preponderante en el diseño de estos protocolos, de ahí que en los últimos años se descubrió que dichos protocolos tienen vulnerabilidades que permiten la manipulación, modificación o intercepción de los datos que transmiten. Por otro lado, existen dispositivos de ICS que tienen varios años o décadas operando en las industrias, por lo que muchos de ellos no cuentan con actualizaciones de seguridad, además son conectados a redes modernas, como Internet, en donde la posibilidad de amenazas es latente. Debido a su gran extensión y a su complejidad en su arquitectura de red y distribución, la probabilidad para encontrar vulnerabilidades en los dispositivos de ICS se incrementa, y la tarea gestionar la red de ICS desde el punto de vista de la seguridad se hace una tarea compleja.

A continuación se presentan algunos de los ataques recopilados en [McLaughlin et al., 2016] y [Pretorius and van Niekerk, 2015] en Sistemas de Control Industrial como ejemplo de la extensión e incremento de este problema de seguridad a lo largo de los años, y de aquí también surge la importancia de su estudio.

- Gasoducto Transiberiano. En 1982 la CIA vulneró con un malware de tipo troyano el sistema de control del gasoducto Transiberiano, causando una explosión equivalente a 3 kilotones de dinamita.

- Sistema de alarma de Chevron. En 1992 un exempleado que había sido despedido por la compañía, deshabilitó la red del sistema de alarmas. No fue detectado hasta que se presentó una emergencia y el sistema de alarmas falló.

- Sistema de agua residuales. En el año 2000, el sistema de control de una planta de tratamiento de aguas residuales en Maroochy Australia, fue vulnerada por un empleado, 800 kilolitros de residuos fueron vertidos en un río.

- Gusano Zotob. En 2005 el gusano Zotob infectó la planta de la compañía automovilística Daimler Chrysler, causando que la producción se detuviera por 50 minutos.

- Ataque Aurora. En 2007 el Laboratorio Nacional de Idaho llevó a cabo el ataque Aurora para demostrar cómo un ciberataque podría causar daño físico a los componentes de la red eléctrica. Una vez que el atacante obtuvo acceso a la red de control de un generador diesel, un malware fue ejecutado para abrir y cerrar rápidamente los interruptores del generador, causando que se desfasara respecto del resto de la red eléctrica, y provocando una explosión en dicho generador.

- Oleducto en Turquía. En 2008 atacantes se infiltraron en el sistema de control del oleoducto Baku–Tbilisi–Ceyhan, lo lograron al vulnerar el software de comunicación de unas cámaras inalámbricas, manipularon dispositivos PLC para que aumentaran la presión en el oleoducto causando una explosión, 30 mil barriles de petróleo fueron vertidos a un manto acuífero. El costo para la compañía British Petroleum fue de 5 millones de dólares por día en aranceles.

- Gusano Stuxnet. En 2009 una planta nuclear de enriquecimiento de uranio en Irán, fue infectada por el gusano Stuxnet, las computadoras infectadas se conectaron a un *Command and control* remoto, a través de éste se reprogramaron los PLC para modificar la operación de las centrifugadoras causando la autodestrucción de éstas.

- Malware Havex. En 2014 se reportó que el malware Havex recolectó datos de sistemas ICS de la industria eléctrica.

- Vulnerabilidad Shellshock. En 2014 se publicó una vulnerabilidad para el sistema de comandos Bash para Linux, la cual permitía la ejecución de código de forma remota, varios sistemas SCADA presentaron esta vulnerabilidad. Aún no se conoce con exactitud el alcance de dispositivos de ICS afectados por esta vulnerabilidad.

Son pocas las industrias del sector público las que han tomado medidas para la seguridad en sus sistemas ICS. En 2016 NERC (North American Electric Reliability Corporation) quien es un organismo internacional cuya misión es garantizar la fiablidad del sistema eléctrico en Norte América [NERC, 2016], por primera vez dictó ordenes para monitorear la continuidad en el servicio eléctrico de la red y el despliegue de defensas para detectar o bloquear malware o comunicaciones maliciosas que afecten la industrial eléctrica [Fairley, 2016]. Con esto se busca que la seguridad en la industria eléctrica se incremente para mitigar los riesgos que esto conlleva.

Es necesaria la atención en la seguridad de los ICS, por ejemplo se estima que las pérdidas para el año 2018 para la industria del petróleo y gas serán de 1.87 millones de dólares [McLaughlin et al., 2016]. De acuerdo al reporte [FireEye, 2016] a partir de la mitad del año 2010 las vulnerabilidades descubiertas en Sistemas de Control Industrial se incrementaron, posiblemente a partir del incidente de seguridad relacionado con el gusano Stuxnet ocurrido en 2009. En la figura 2-3 resultado de un estudio realizado por la compañía FireEye en donde se analizaron 1552 vulnerabilidades en productos de ICS de 123 fabricantes entre los años

2000 y 2015, se aprecia que más del 90 % de las vulnerabilidades descubiertas se presentaron después de 2010, aunque se cree también que esta cifra se incrementó respecto a los años anteriores debido a que en noviembre de 2009 entró en operaciones el Equipo de Respuestas a Incidentes en ICS (ICS-CERT) con lo cual se reportaron una mayor cantidad de incidentes de ICS; se logra apreciar otro incremento importante entre 2014 y 2015 del 49 %.

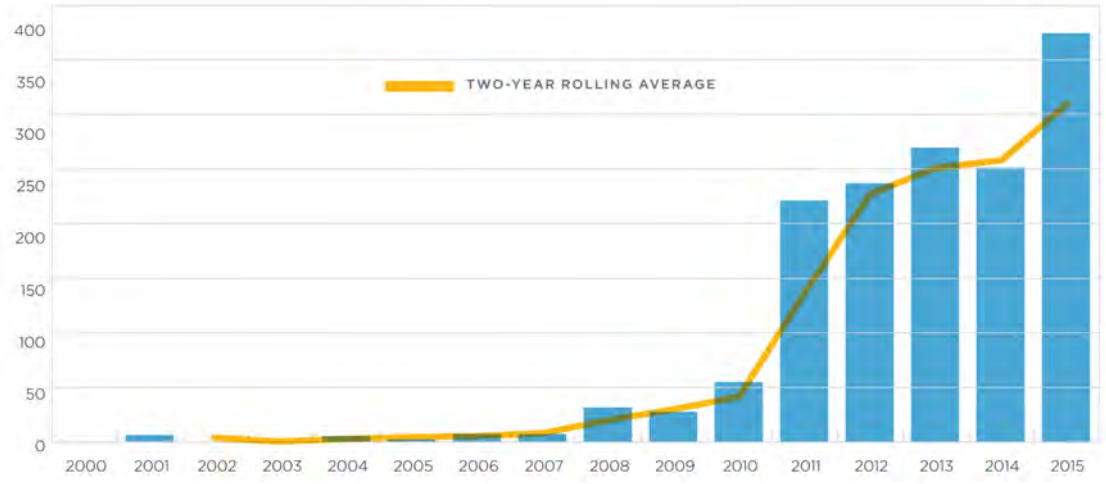


Figura 2-3: Vulnerabilidades descubiertas en productos específicos por año. [FireEye, 2016]

El Modelo jerárquico de Purdue para ICS, desarrollado por la Sociedad Internacional de Automatización ISA-99, divide en 5 zonas y 6 niveles de operación a una infraestructura de ICS . Usa el concepto de zonas para subdividir una red Corporativa y de ICS en segmentos lógicos compuestos por sistemas que llevan a cabo funciones similares o que tienen requerimientos similares [Obregon, 2015]. En el mismo reporte de FireEye mencionado con anterioridad se muestra un diagrama simplificado de este Modelo, y se presenta en la figura 2-4.

En la Zona 0 se encuentran los sensores y actuadores que interactúan directamente con el proceso industrial, estos miden por ejemplo temperatura o presión, abren y cierran válvulas, encienden o apagan motores o bombas. En la Zona 1 se encuentran los dispositivos como PLC o RTU, envían y reciben información desde y para los dispositivos de la Zona 0. En la Zona 2 se encuentran los dispositivos que permiten a un operador realizar labores de supervisión y control del proceso industrial. En la Zona 3 es posible tener acceso a datos y aplicaciones que permiten acceder a la red de control remotamente. La Zona 4 es la red corporativa, en donde se encuentran los servidores de correo electrónico, bases de datos, impresoras, etc. En la Zona 5 se encuentran la red pública como Internet, desde la cual se podría acceder remotamente al sistema ICS.

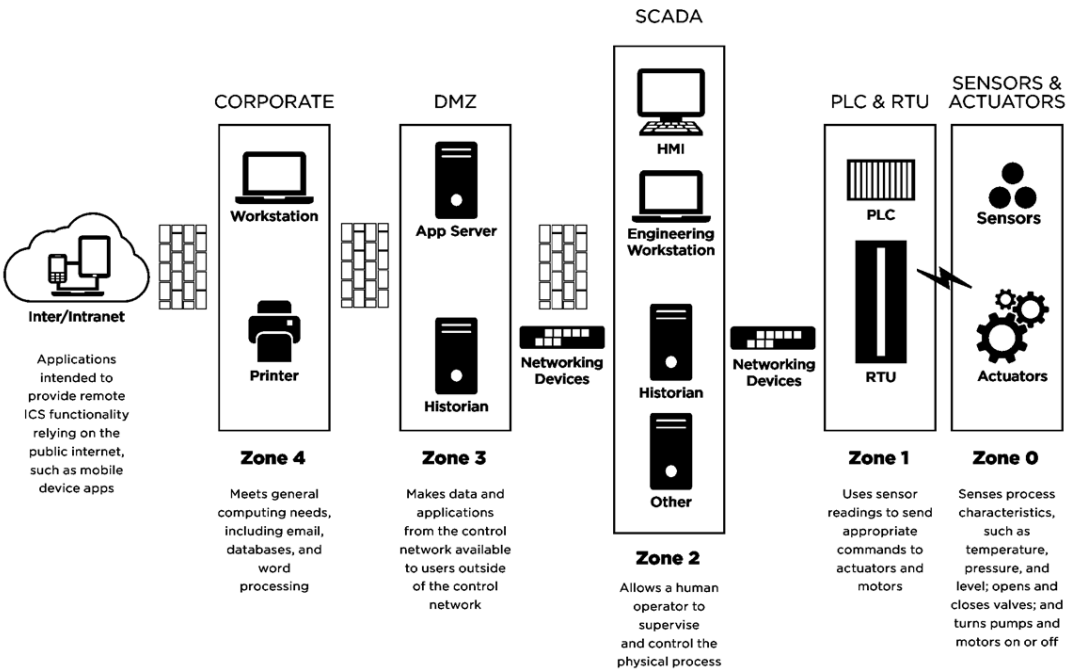


Figura 2-4: Modelo simplificado de Purdue de un ICS. [FireEye, 2016]

Cerca de la mitad de las vulnerabilidades descubiertas entre febrero de 2013 y abril de 2016 afectaron a dispositivos de la Zona 2, en segundo lugar de estas afectaciones se encuentran los dispositivos pertenecientes a la Zona 1, de acuerdo al mismo reporte de FireEye, en la figura 2-5 se presentan estos datos. El protocolo de comunicación Modbus que se estudia en esta tesis, se usa en la comunicación entre los dispositivos de la Zona 1 y 2, y precisamente es en donde se presentaron más vulnerabilidades de acuerdo a este reporte.

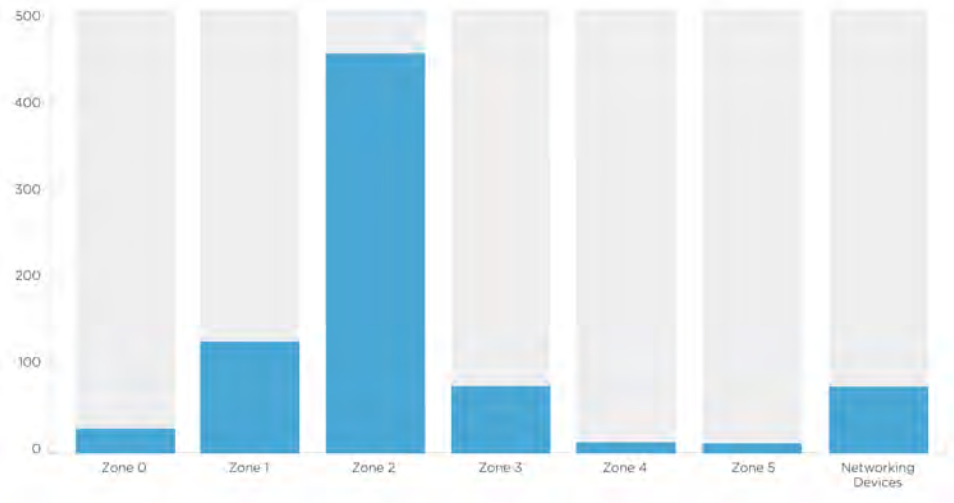


Figura 2-5: Vulnerabilidades por Zonas, descubiertas entre febrero de 2013 y abril de 2016. [FireEye, 2016]

De acuerdo a NIST [2], los ICS son diferentes de los sistemas comunes por lo tanto las soluciones de seguridad tradicionales presentan fallas para asegurarlas, estas diferencias son presentadas en [McLaughlin et al., 2016] y se listan a continuación;

- El objetivo principal de un ICS es mantener la operación del proceso industrial.
- Los procesos industriales necesitan tener alta disponibilidad, cualquier reparación necesita ser planificada.
- La interacción con el proceso físico es fundamental y muchas veces complejo.
- Los recursos se centran en preservar la integridad del proceso industrial y no en aspectos de seguridad.
- El tiempo de respuesta oportuna ante una eventualidad en los sensores es crítica.
- Los ICS suelen utilizar protocolos propietarios para controlar a los dispositivos remotos.
- Los componentes de ICS no se reemplazan frecuentemente, por ejemplo cada 15 o 20 años.
- La ubicación de los componentes de ICS dificultan su acceso para reparaciones y actualizaciones.

En el año 2008 la marca Tofino sacó a la venta su solución de seguridad para sistemas SCADA (Tofino Industrial Security Solution), este firewall verificaba la validez de cada paquete SCADA en la red, no sólo revisando qué protocolos estaban permitidos sino también verificando la funcionalidad del mensaje, por ejemplo la funcionalidad entre un mensaje de sólo lectura de un dispositivo y uno en el que se reprograma el funcionamiento del dispositivo. Recientemente han surgido herramientas para la detección de anomalias en el tráfico de ICS basadas en patrones de tráfico, algunas de ellas pertenecen a las compañías Dragos Security, o NexDefense [Fairley, 2016]. Las compañías Tofino/Belden, Secure Crossing, ScadaFence y SilentDefense, entre otras, ofrecen sistemas de seguridad para protocolos ICS, tanto para protocolos abiertos como Modbus o DNP3, como para protocolos propietarios de sistemas específicos [Knapp and Langill, 2015]. Por otro lado, desde 2012 se cuentan con preprocesadores en Snort[3] para los protocolos DNP3 y Modbus, se pueden usar a partir de la versión 2.9.2 [Combs, 2012]. A pesar de estos esfuerzos aún no se han generalizado eficientemente el uso de tecnologías para la seguridad de los Sistemas de Control Industrial.

[2]National Institute of Standards and Technology, es una agencia del gobierno de EE.UU. que se encarga de promover el avance tecnológico y de dictar normas y estándares para la industria de diversas tecnologías.

[3]Software especializado para la detección de intrusos en el tráfico de red, es un IDS de red.

Capítulo 3

Protocolo Modbus TCP/IP

3.1. Resumen

En este capítulo se detallan los conceptos asociados al protocolo Modbus para TCP/IP. Se revisan conceptos de redes de computadoras como el Modelo TCP/IP y el modelo cliente-servidor; para finalmente detallarse el protocolo de aplicación Modbus. Se explica a fondo un ejemplo de la comunicación entre dos dispositivos que utilizan al protocolo Modbus TCP/IP. Se da la interpretación del funcionamiento del circuito electrónico que provee los datos y del dispositivo electrónico que los adquiere, denominado esclavo, y de que manera se establece la comunicación de red con el dispositivo que los recibirá, denominado maestro, haciendo el estudio correspondiente del tráfico de red. Se estudia el protocolo Modbus TCP/IP con el fin de, en un análisis posterior, detectar cuales son las vulnerabilidades a las que es susceptible, y para la interpretación de los resultados de la detección de anomalías.

3.2. Origen del protocolo Modbus

El protocolo de comunicación para SCADA Modbus fue desarrollado en 1970, por la compañía Modicon, es ampliamente usado para la interconexión de dispositivos en sistemas industriales, es un protocolo libre y de código abierto [Devarajan, 2007]. Cada tipo de dispositivo como PLC (Programmable Logic Controller), HMI (Human-Machine Interface), Controladores, Dispositivos de entrada/salida entre otros, pueden usar el protocolo Modbus para llevar a cabo una operación remota. La misma comunicación se puede llevar a cabo tanto de forma serial (Modbus Serial) como a través de una red Ethernet TCP/IP (Modbus TCP/IP). Las puertas de enlace permiten la comunicación entre varios tipos de conexiones o redes usando el protocolo Modbus. Es posible llevar a cabo esta conexión entre tecnologías diferentes porque Modbus define una unidad de datos de protocolo (Protocol Data Unit, PDU) independiente de las capas inferiores en el protocolo de comunicación usado. Para mapear el protocolo Modbus en conexiones o redes específicas se pueden agregar campos adicionales a la unidad de datos de la capa de aplicación (ADU) [Modbus Organization, 2006], tal es el caso de Modbus TCP/IP. Los tres formatos ADU estándares son RTU (Remote Terminal Unit), ASCII, generalmente usados en forma serial y TCP/IP en redes ethernet [National Instruments, 2014].

En 1979 fue introducido al mercado Modbus para conexiones de tipo serial. En 1999 Modicon liberó una especificación de Modbus para su uso con los protocolos de TCP/IP. En

la práctica es el protocolo más utilizado en la industria de control, es decir, es el estándar *de facto* para los ICS [Jiménez Díaz, 2011]. Modbus TCP/IP es un protocolo utilizado para la comunicación entre PLC, computadoras, sensores y otros dispositivos físicos de entrada y/o salida, sobre el protocolo TCP/IP.

3.3. El Modelo TCP/IP

El modelo TCP/IP es un conjunto de protocolos que interactúan entre sí para poder lograr la comunicación entre dos o más dispositivos de cómputo a través de una red. Está divido en cuatro capas, cada una de ellas es responsable de una fase en la comunicación. A continuación de describe cada una de las capas.

- Aplicación. Esta capa se encarga de tratar los datos de la aplicación tanto en el dispositivo que envía como en el que recibe, se encarga de la forma de presentar la información, de cómo es codificada, algunos protocolos de esta capa son HTTP, DNS, SMTP, DHCP, Modbus.
- Transporte. Esta capa establece algunos aspectos de la forma de comunicación entre dos dispositivos, provee de transporte a los datos que irán a la capa de aplicación. Provee un nivel de confiabilidad en la comunicación sobre las otras capas inferiores que no lo hacen. Los dos protocolos ampliamente utilizados en las comunicaciones para esta capa son TCP (Transmission Control Protocol) y UDP (User Datagram Protocol). Una forma de implementación del protocolo Modbus utiliza a TCP como protocolo de transporte.
- Red. Es la capa responsable de dirigir los datos del dispositivo destino al origen a través de redes diferentes. Establece un sistema de direcciones lógicas para lograr esta comunicación.
- Enlace. Se encarga de las comunicaciones desde un dispositivo hacia el medio físico en el cual reside. Trata el envío y recepción de datos de un dispositivo sobre una determinada interfaz de comunicación hacia la red. En esta capa se lleva a cabo la traducción entre direcciones lógicas (direcciones IP) a direcciones físicas (direcciones MAC). Por ejemplo en esta capa se usa el estándar Ethernet.

El modelo TCP/IP tiene una correspondencia con el modelo de referencia OSI (Open Systems Interconnection), mostrada en la figura 3-1, que se usa como base para establecer estándares que permitan la comunicación entre sistemas diferentes.

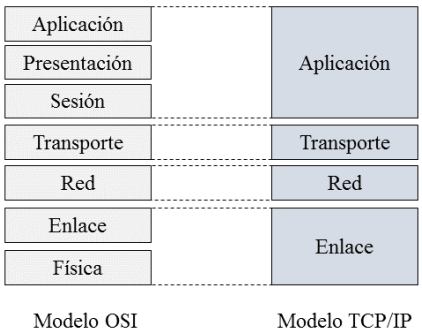


Figura 3-1: Correspondencia entre los modelos OSI y TCP/IP.

En la figura 3-2 se muestra la interacción entre dos dispositivos que utilizan a Modbus como protocolo de comunicación bajo el modelo TCP/IP. El proceso inicia cuando el maestro o cliente realiza una conexión hacia el esclavo o servidor, el sistema operativo del maestro agrega los datos necesarios a esa petición para que pueda llegar a su destino en el dispositivo esclavo. La información viaja desde la red del maestro hacia la red del esclavo, al llegar a este último obtiene los datos enviados por el dispositivo maestro con el fin de contestar la solicitud. Es posible que tanto el maestro como el esclavo se encuentren en la misma red, si es así, no será necesario el uso de un router para la conexión de ambas redes.

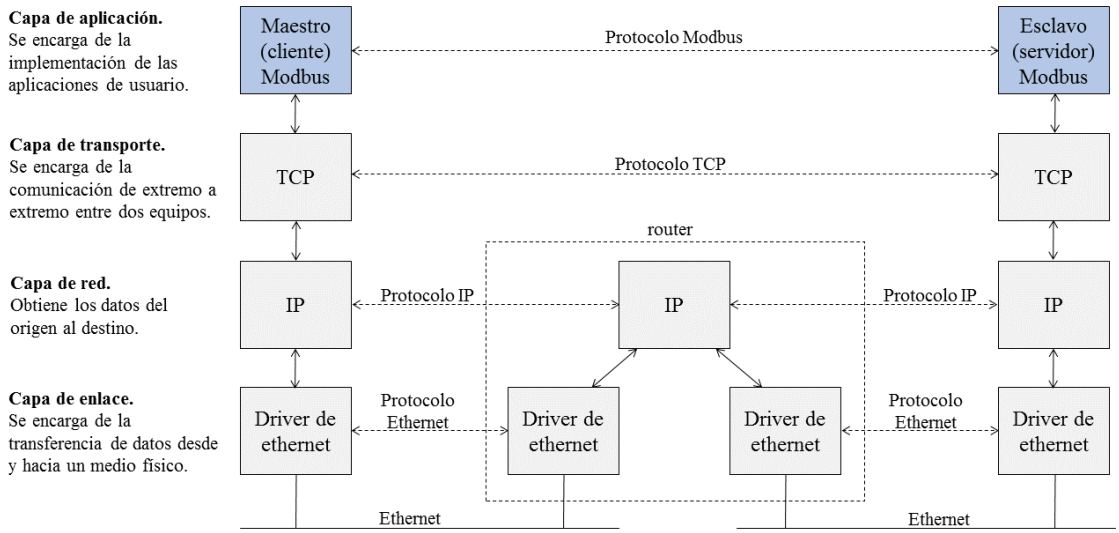


Figura 3-2: Comunicación entre dos dispositivos a través del protocolo Modbus, usando el modelo TCP/IP. [modificada de Fall and Stevens, 2012]

3.4. Modelo cliente-servidor y Modbus TCP/IP

Las conexiones entre el maestro y esclavo en Modbus siguen el modelo cliente-servidor sobre una red TCP/IP. Los mensajes entre maestro y esclavo, o cliente y servidor, son usados para intercambiar información de control en tiempo real. Los mensajes son de solicitud o *Request* y de respuesta o *Response*:

- Request. Es el mensaje de solicitud que envía el maestro o cliente hacia el esclavo o servidor, para iniciar una transacción.
- Response. Es el mensaje de respuesta enviado por el esclavo o servidor, previa solicitud del maestro o cliente.

Únicamente el maestro puede iniciar las solicitudes o *Request* de lectura y/o escritura a los esclavos, enviado un paquetes a éstos. Es importante aclarar que el esclavo es el que actúa como servidor, esto se debe a que recibe todas las solicitudes del maestro, de lectura o de escritura, estas solicitudes son para obtener o dar instrucciones a dispositivos como válvulas, medidores de presión, motores, etc, el esclavo entonces procesa la solicitud y dependiendo del dispositivo envía un paquete de respuesta o *Response*. Por lo general el esclavo es un controlador lógico programable (PLC, *Programmable Logic Controller*) o un controlador de

automatización programable (PAC, *Programmable Automation Controller*). Se ha establecido que los esclavos reciban las peticiones provenientes del maestro en el puerto 502 [Internet Assigned Numbers Authority, 2016], sin embargo y dependiendo del dispositivo, es posible cambiar este puerto; por otro lado, el maestro toma el papel de cliente al realizar las peticiones de conexión al esclavo, por lo general el maestro es una interfaz HMI o parte de un sistema SCADA. En la figura 3-3 se muestra un diagrama de este esquema cliente-servidor.

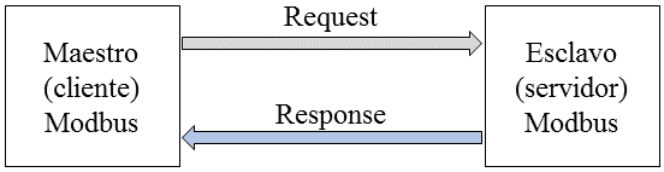


Figura 3-3: Modelo cliente-servidor para Modbus

3.5. Paquetes Modbus TCP/IP

El protocolo Modbus, que se encuentra en la capa de aplicación del protocolo TCP/IP, tiene una estructura dividida en dos partes, la cabecera Modbus Application Protocol o MBAP Header; y los datos o Modbus Application PDU (Protocol Data Unit). Cada paquete de Modbus tiene una estructura definida en [Modbus Organization, 2006], que se detalla a continuación.

La cabecera Modbus Application Protocol (MBAP header), cuenta con cuatro campos:

1. Identificador de transacción o *Transaction Identifier* (2 bytes). Se usa para sincronizar los mensajes entre el maestro y el esclavo, es decir, permite a ambos llevar un control para asociar a cada solicitud o *Request* con su respuesta o *Response*, previo establecimiento de la comunicación. El servidor usa el mismo identificador de transacción en la respuesta que el recibido en la solicitud enviada por el cliente. En una conexión TCP, este identificador debe ser único.

2. Identificador de Protocolo o *Protocol Identifier* (2 bytes). Es usado para la multiplexación entre sistemas[1]. El protocolo Modbus es identificado por el valor 0.

3. Tamaño del paquete o *Length field* (2 bytes). Contiene el número de bytes que restan en el paquete, incluyendo el Identificador de unidad y los campos de los datos o PDU.

4. Identificador de Unidad o *Unit identifier* (1 byte). Este campo se usa para identificar al dispositivo esclavo que utiliza Modbus+ o Modbus serial. Para Modbus TCP/IP el identificador del esclavo corresponde a la dirección IP, por lo que este valor es 0xFF.

Después de la MBAP Header viene el Modbus Application Protocol Data Unit (PDU), y está formado:

1. Código de función o *Function code* (1 byte). Son las funciones soportadas por Modbus.

2. Datos asociados al tipo de función o *Functions Data* (n bytes). Son los datos que corresponden a una función de Modbus.

[1] conocido como *intra-system multiplexing*, se refiere a cuando un mismo medio es utilizado por varios tipos de sistemas.

En el PDU se define un código de función (*Function code*), parámetros asociados a la función definida por el código, y por los datos propios de dicha función. Estos códigos especifican el tipo de operación que llevará a cabo el esclavo, por ejemplo monitorear, configurar o controlar los módulos de entrada y/o salida del esclavo. Estos módulos de entrada y/o salida están conectados a válvulas, actuadores, sensores, medidores de temperatura o presión, motores, etc, es decir, estos códigos dan instrucciones a los esclavos para el acceso y manipulación de datos. Existen códigos predeterminados en la definición del protocolo [Modbus Organization, 2006], sin embargo cada fabricante podría realizar sus propias definiciones, implementándolas en sus dispositivos esclavos y maestros. La PDU tiene un máximo tamaño de 253 bytes y la longitud de la MBAP header es un valor fijo de 7 bytes, por lo que el máximo tamaño de los datos de aplicación de TCP es de 260 bytes [Organization, 2012]. En la figura 3-4 se muestra como se conforma el Modbus TCP/IP ADU.

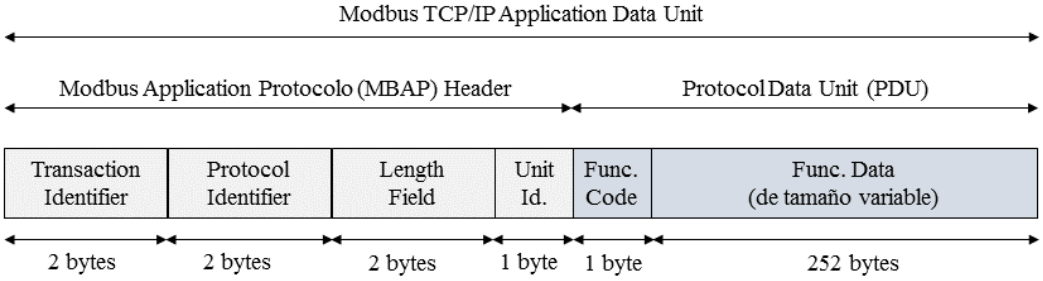


Figura 3-4: Modbus TCP/IP Application Data Unit

En la figura 3-5 se muestra de forma general como es la arquitectua de conexión entre maestro y esclavos en una red ethernet, además se muestra el encapsulado de protocolos de la pila TCP/IP usando el protocolo Modbus.

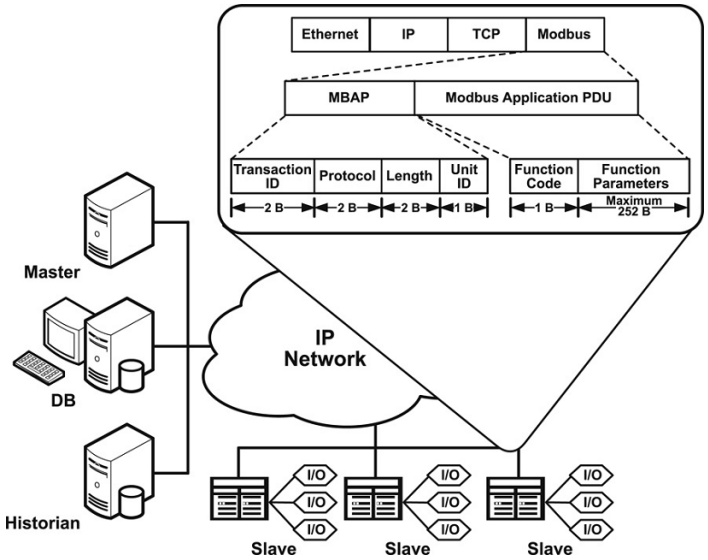


Figura 3-5: Arquitectura Modbus para TCP/IP [Jiménez Díaz, 2011]

Cuando el cliente envía al servidor un paquete *Request*, es la función contenida en este paquete la que le indica al servidor que tipo de acción debe llevar a cabo. Es el campo de Datos asociados al tipo de función el que contiene la información adicional necesaria para

que el servidor lleve a cabo la acción en base al código de función, ambos son dados por el cliente al momento del envío del *Request*. Estos datos adicionales pueden incluir valores como direcciones de registros, valores discretos, la cantidad de valores, y el tamaño de los valores (en bytes) que se enviarán. Existen algunas funciones para las que no es necesario que el servidor reciba más datos que el código de función, es decir el código de función define por si mismo la acción que debe llevar a cabo el servidor. Si no ocurren errores relacionados con la función solicitada, el servidor responderá con un paquete *Response* que contenga el mismo código de función y los datos resultantes de la acción llevada a cabo [Organization, 2012]. En la figura 3-6 se ilustra una transacción Modbus.

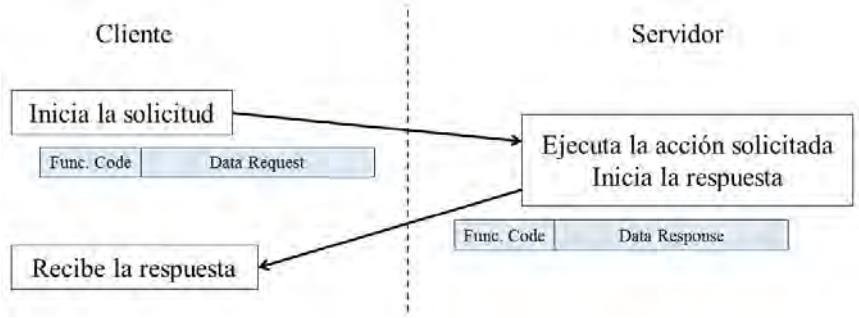


Figura 3-6: Transacción Modbus sin error [Organization, 2012]

Si ocurre un error relacionado con la función de Modbus solicitada, el servidor envía un paquete *Response* con un código de excepción, el cual indica la razón del error con el fin de dar información relevante al cliente del error detectado durante el procesamiento de la acción por parte del servidor, estos errores pueden ser desde una solicitud malformada hasta solicitudes incorrectas. Sin embargo, las excepciones también se pueden generar como una respuesta a nivel de la aplicación para una solicitud válida. El PDU de una excepción tiene una forma definida, el código de función que se regresa al maestro que envió un paquete *Request* es igual al código de función recibido con el conjunto de bits más significativo activo, es decir, se le suma el valor 0x80 al código de función enviado por el cliente. A diferencia de los datos asociados a los códigos de función normales, las excepciones no requieren de datos adicionales [National Instruments, 2014]. En la figura 3-7 se ilustra una transacción Modbus con una excepción como respuesta [Organization, 2012].

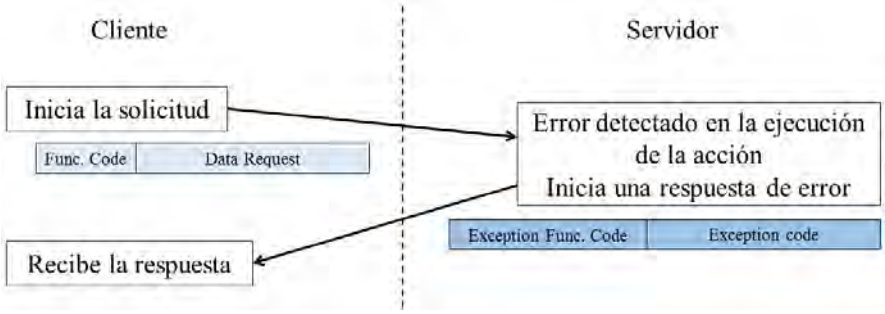


Figura 3-7: Transacción Modbus con una excepción como respuesta [Organization, 2012]

Existen tres diferentes PDU para el protocolo Modbus, las cuales se muestran en la siguiente tabla 3.1 [Organization, 2012].

Tipo de PDU	Código de función		Datos de la función	
	Tamaño	Descripción	Tamaño	Descripción de los datos
Request	1 byte	Código de función.	n bytes	Los valores de los datos dependen del tipo de función.
Response	1 byte	Mismo código de función que el Request recibido.	n bytes	Los valores de los datos dependen del tipo de función.
Exception response	1 byte	Mismo código de función que el Request recibido + 0x80	1 byte	Código de excepción de Modbus, generalmente 01, 02, 03 o 04.

Tabla 3.1: Tipos de PDU en Modbus.

Modbus utiliza el formato *big endian* para representar la información, cuando un valor mayor a un byte es transmitido, el bit más significativo es enviado primero. Por ejemplo, si se requiere enviar el valor 0x1234, el primer byte a enviar es 0x12 seguido de 0x34 [Organization, 2012]. Este mismo formato es utilizado por el protocolo TCP/IP para la transmisión de datos [RFC791, 1981].

3.6. Modelo de datos de Modbus

En el protocolo Modbus existen bloques o rangos de dirección que se usan para almacenar los datos y son, entradas discretas (*Discretes input*), salidas digitales, salidas discretas o bobinas (*Coils* o *discrete outputs*), registros de entrada (*Input registers*) y registros de retención, registros internos o registros de salida (*Holding registers* o *output registers*). Estos bloques definen el tipo y los permisos de accesos a los datos. Los esclavos tienen el acceso a estos datos puesto que la comunicación con el dipositivo de control es directo, y son alojados localmente. Estos datos por lo general residen en una parte de la memoria principal[2] del esclavo, los maestros entonces, solicitan el acceso a estos datos a través de los códigos de función. Estos datos al momento de enviarse del esclavo al maestro forman parte de los datos asociados a la función dentro de Modbus PDU. El modelo de bloques solamente es conceptual, es decir, cada esclavo define la forma de organización y acceso de los datos dentro de su memoria principal [National Instruments, 2014]. Estos bloques se detallan en la tabla 3.2.

Tablas primarias	Tipo de objeto	Tipo	Descripción
Discretes input	bit	lectura	Este tipo de datos es dado por un sistema de entrada/salida (I/O).
Coils	bit	lectura - escritura	Este tipo de datos pueden ser modificados por un programa.
Input registers	16 bit (una palabra)	lectura	Este tipo de datos es dado por un sistema de entrada/salida (I/O).
Holding registers	16 bit (una palabra)	lectura - escritura	Este tipo de datos pueden ser modificados por un programa.

Tabla 3.2: Modelo de bloques de almacenamiento de datos en Modbus [Organization, 2012]

[2]memoria RAM

3.7. Funciones de Modbus

Las funciones en Modbus permiten el acceso por parte del maestro a los bloques de datos del esclavo, ya sean funciones de lectura o escritura, una forma de llevar a cabo esto es que a través de las funciones el maestro tenga acceso a las ubicaciones estáticas de la memoria, aunque esta forma puede variar dependiendo de la implementación del protocolo Modbus. Por ejemplo, un código de función 1, (leer entradas discretas o *Coils*) y un código de función 3 (leer registros de salida o *Holding registers*), pueden tener acceso a la misma ubicación física en la memoria, pero podría ser que un código de función 3 y un código de función 16 (escribir a registros de salida o *Holding registers*), tengan acceso a ubicaciones diferentes en la memoria. La forma en cómo es ejecutado cada código de función es definida por la implementación del modelo de datos propio del esclavo. [National Instruments, 2014].

De acuerdo a [Organization, 2012] existen tres categorías de códigos de función:

- Públicos. Son códigos de funció definidos por la organización Modbus, existe amplia documentación al respecto, incluidas pruebas de su funcionamiento. Incluye tanto funciones definidas como códigos de función reservados para uso futuro.
- Definidos por el usuario. Existen dos rangos de códigos de funciones, del 65 al 72 y del 100 al 110. Cada usuario puede diseñar sus propia implementación de funciones.
- Reservados. Estos códigos de función son utilizados en productos que ya no reciben soporte de algunas compañías y no están disponibles para su uso.

En la siguiente tabla 3.3 se listan algunos de los códigos de función públicos definidos por la organización Modbus, se especifica el tipo de acceso a los datos, ya sea que se acceda a un sólo bit o a registros de 16 bits, además se muestran algunos códigos de diagnóstico, los cuales se utilizan para recabar datos de diagnóstico o información de control del dispositivo esclavo.

Tipo			**Función**	**Códigos de función**	
				Código	**Subcódigo**
Acceso a los datos	Acesso por bit	Entradas físicas discretas	Read discrete inputs	02	
		Bits internos o salidas físicas discretas	Read coils	01	
			Write single coil	05	
			Write multiple coils	15	
	Acceso por 16 bits	Registros de entrada físicos	Read input register	04	
		Registros internos o registros físicos de salida	Read holding registers	03	
			Write single register	06	
			Write multiple registers	16	
			Read/write multiple registers	23	
			Mask write register	22	
			Read FIFO queue	24	
Diagnóstico			Read exception status	07	
			Diagnostic	08	00-18,20
			Report server ID	17	
			Read device identification	43	14

Tabla 3.3: Extracto de los códigos de función públicos.

A continuación se describen a modo de ejemplo las funciones Lectura de registros de salida o *Read Holding registers*, y Escritura de múltiples entradas discretas o *Write multiple coils*.

Función *Read Holding registers.*
Esta función permite leer el contenido de bloques de datos contiguos de los registros de salida o *Holding registers*, de un dispositivo esclavo. El paquete *Request* especifica el inicio de la dirección del registro y el número de registros de salida. Los registros empiezan en cero, y son direccionados de 0 a 15, no 1 a 16. En el paquete Response, los datos de respuesta están ordenados en 2 bytes por registro, para cada registro el primer byte contiene los bits más significativos y el segundo byte los bits menos significativos. En las tablas 3.4, 3.5 y 3.6 se muestran la forma en que están conformadas las PDU de los paquetes *Request*, *Response* y de error o excepción como respuesta, respectivamente ($^*N$ = cantidad de registros).

Código de función	1 byte	0x03
Dirección inicial	2 bytes	0x0000 a 0xFFFF
Cantidad de salidas	2 bytes	0x0001 a 0x7D

Tabla 3.4: PDU del *Request* para el código de función 3.

Código de función	1 byte	0x03
Contador de bytes	1 bytes	$2 \times N^*$
Valor del registro	$N^* \times 2$ bytes	

Tabla 3.5: PDU del *Response* para el código de función 3.

Código de error	1 byte	0x83
Código de excepción	1 byte	01, 02, 03 o 04

Tabla 3.6: PDU de *Excepción* para el código de función 3.

Función *Write Multiple Coils.*
Esta función permite forzar a cada salida discreta o *Coil*, dentro de una secuencia de *Coils*, a tener el valor de 1 o 0 (*on* u *off*) en un dispositivo esclavo. La PDU del *Request* indica la referencia del *Coil* que será forzado a tener un valor. Los *Coils* son direccionados empezando desde cero. El paquete *Response* regresará además del mismo código de función, la dirección de inicio, y la cantidad de *Coils* que fueron forzados a tener un valor. En las tablas 3.7, 3.8 y 3.9 se muestran la forma en que están conformados las PDU de los paquetes *Request*, *Response* y de error o excepción como respuesta, respectivamente ($^*N$ = *Cantidad de salidas* /8, si el residuo es diferente de cero entonces, $N = N + 1$).

Código de función	1 byte	0x0F
Dirección inicial	2 bytes	0x0000 a 0xFFFF
Cantidad de salidas	2 bytes	0x0001 a 0x07B0
Contador de bytes	1 byte	$N*$
Valores de los datos	$N* \times 1$ byte	

Tabla 3.7: PDU del *Request* para el código de función 15.

Código de función	1 byte	0x0F
Dirección inicial	2 bytes	0x0000 a 0xFFFF
Cantidad de salidas	2 bytes	0x0001 a 07B0

Tabla 3.8: PDU del *Response* para el código de función 15.

Código de error	1 byte	0x8F
Código de excepción	1 byte	01, 02, 03 o 04

Tabla 3.9: PDU de *Excepción* para el código de función 15.

3.8. Ejemplo de una comunicación de Modbus TCP/IP

A continuación se describe un ejemplo de una comunicación a través del protocolo Modbus TCP/IP entre un maestro y un esclavo. Se muestra el tráfico de red, desde el establecimiento de la conexión a través del *Three-way handshake*, hasta el intercambio de paquetes siguiendo el protocolo Modbus. Además se da la interpretación asociada al dispositivo electrónico que se utilizó.

3.8.1. Análisis del Three-way handshake

Ray Tomlinson introdujo en 1975 el concepto del three-way handshake [Tanenbaum and Wetherall, 2012], el cual es un protocolo de establecimiento de comunicación para la transmisión de datos. Este proceso se realiza generalmente antes de que cualquier dato sea enviado entre los equipos. Lo que se representa, es un cliente o equipo origen iniciando una conexión al servidor o equipo destino. Es necesario para la comunicación usando Modbus TCP/IP el establecimiento de una conexión TCP entre un cliente y el servidor, esto se logra a través del *Three-way handshake*; dicha comunicación es llevada a cabo al enviar y recibir los datos a través de *sockets*, para esto es necesario que un puerto identifique al servicio ofrecido por Modbus por parte del servidor, mientras que el cliente deberá usar un puerto mayor a 1024, un puerto diferente para cada conexión de tipo cliente.

Cada equipo de la sesión seleccionará un número de secuencia inicial (ISN por sus siglas en inglés) como el primer número de secuencia. Esto significa que el cliente y el servidor seleccionarán diferentes números de secuencia individuales. La generación de estos números de secuencia depende de la implementación de la pila TCP/IP de cada sistema operativo. Es conveniente que estos números sean generados de manera aleatoria, de manera que no puedan ser adivinados, porque la seguridad de las sesiones podría verse afectada [Northcutt and Novak, 2002].

Estos números de secuencia proporcionan el mecanismo de ordenamiento de paquetes para el flujo de datos TCP. Desde que el flujo es iniciado con un número de secuencia inicial, un paquete perdido es fácil de identificar. También, si un paquete TCP llega al equipo destino en un orden diferente al que fue enviado, el equipo destino puede ordenarlo de la forma correcta a través del número de secuencia [Fall and Stevens, 2012]. En la figura 3-8 se ilustra este proceso.

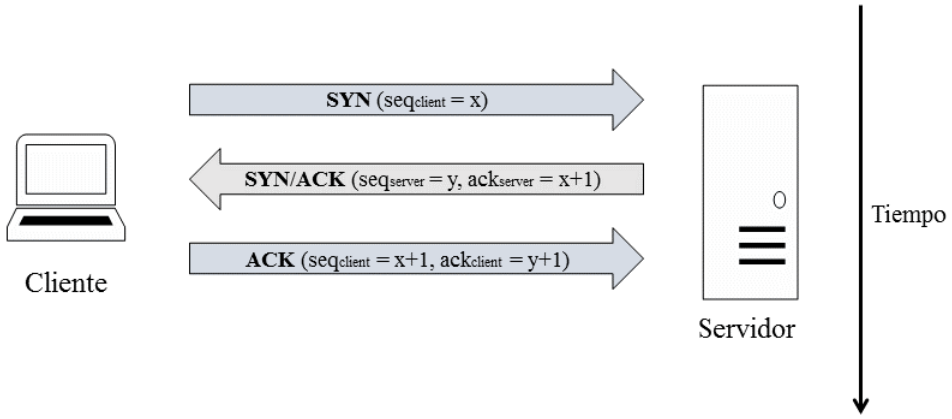


Figura 3-8: Three-way handshake

A continuación se muestra el *Three-way handshake* en el tráfico de red, los tres paquetes corresponden al inicio de la comunicación entre un dispositivo maestro y uno esclavo que utilizan al protocolo de aplicación Modbus. La forma de presentar los datos corresponde al analizador de protocolos Tcpdump. Es importante señalar que los analizadores de protocolos como Tcpdump/Windump o Wireshark, de forma predeterminada pueden llegar a desplegar números de secuencia y de confirmación "relativos" en lugar de los verdaderos valores de cada paquete, esto lo realizan para que la lectura de los paquetes durante el análisis sea de mayor facilidad. Los datos marcados con color verde corresponden al protocolo IP, mientras que los datos marcados con color azul corresponden al protocolo TCP, aquí se puede apreciar el concepto de encapsulado de protocolos en la pila TCP/IP.

```
18:12:44.069149 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [S], seq 374218748,
win 29200, options [mss 1460,sackOK,TS val 242176
       0x0000: 4500 003c adfc 4000 4006 08b2 c0a8 0145 E..<..@.@......E
       0x0010: c0a8 0178 9c85 01f6 164e 1ffc 0000 0000 ...x.....N......
       0x0020: a002 7210 843c 0000 0204 05b4 0402 080a ..r..<..........
       0x0030: 0003 b200 0000 0000 0103 0306           ............
18:12:44.069570 IP 192.168.1.120.502 >192.168.1.69.40069: Flags [S.], seq 4107683793,
ack 374218749, win 2048, options [mss 1460], length 0
       0x0000: 4500 002c 0ec1 4000 8006 67fd c0a8 0178 E..,..@...g....x
       0x0010: c0a8 0145 01f6 9c85 f4d6 47d1 164e 1ffd ...E......G..N..
       0x0020: 6012 0800 fa99 0000 0204 05b4 0000      `.............
18:12:44.070411 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [.], ack 1, win 29200,
length 0
       0x0000: 4500 0028 adfd 4000 4006 08c5 c0a8 0145 E..(..@.@......E
       0x0010: c0a8 0178 9c85 01f6 164e 1ffd f4d6 47d2 .Q.V.u..........
       0x0020: 5010 7210 8428 0000                     P.r..(..
```

El primer paquete corresponde al inicio de conexión del dispositivo con dirección IP 192.168.1.69, el maestro, éste le envía un paquete de sincronización (con la bandera SYN activada) al dispositivo con la dirección IP 192.168.1.120, el esclavo, el servicio que ofrece este último corresponde al protocolo Modbus, al puerto 502. El esclavo responde al maestro con un paquete de confirmación (con la bandera ACK activada), y en el mismo paquete le envía un mensaje de sincronización (con la bandera SYN activada). Por último en el tercer paquete, el maestro confirma la petición de sincronización del esclavo (con la bandera ACK).

Siguiendo el modelo cliente-servidor, se aprecia que el cliente o maestro, es quien solicita la comunicación, a través de un puerto mayor al 1024, al servidor o esclavo en el puerto 502, que corresponde como ya se mencionó al protocolo Modbus. Es importante remarcar que el servidor o esclavo Modbus no cuenta con un sistema de autenticación, es decir, cualquier dispositivo cliente o maestro, puede enviar solicitudes de conexión al esclavo, si éste esta ofreciendo el servicio podrá establecer la comunicación sin conocer si es una petición legítima. Es decir, si se quiere establecer una autenticación con el esclavo será necesario hacer uso de otros dispositivos de red o incluso diseñar algún protocolo para solicitar esto. Además la comunicación en ningún momento va cifrada, si se requiere añadir este servicio de seguridad será necesario realizarlo con algún otro dispositivo de cómputo o protocolo adicional.

3.8.2. Análisis de Request-Response

Después de que la comunicación se ha establecido entre el maestro y el esclavo, el primero comenzará a solicitarle o enviarle información al segundo, el protocolo Modbus tiene un sistema simple para realizar estas funciones. El maestro envía al esclavo un paquete Modbus de solicitud o envío de información o *Request* y el esclavo responderá a tal solicitud con un paquete Modbus de respuesta o *Response*, es decir, para cada paquete *Request* del maestro, habrá un paquete *Response* por parte del esclavo, esto es una transacción de Modbus.

El dispositivo esclavo de este ejemplo recibe el valor de luminosidad de una fotoresistencia de un circuito eléctrico, y dependiendo del valor de ésta envía valores de voltaje al circuito para encender uno, dos o tres led[3], mientras mayor sea el valor de luminosidad mayor el número de led encendidos. A cada led se le asigna una salida discreta o *Coil*, un valor de 1 significa que el led debe encender, un valor de 0 significa que el led debe permanecer apagado, es decir, son valores de un bit por cada salida discreta, un bit por cada led. Si se requiere encender el 1er y 2o led, se tendrá que enviar la secuencia de bits 011, se usan tres bits puesto que son tres led. Es necesario completar o rellenar con bits para tener un byte, entonces la secuencia de bits es 00000011, y en formato hexadecimal es igual a 0x03, estos datos son los que el maestro deberá a enviar al esclavo, previo establecimiento de la conexón, para encender dichos led. En la figura B-1 se muestra la configuración del PLC utilizado así como los datos que puede enviar el maestro al esclavo, dichos datos llegarán al esclavo, es decir, el maestro envía datos para guardar o escribir en la memoria del esclavo, de esta forma el esclavo enviara estos datos a través de sus salidas hacia el circuito eléctrico.

Datos en Modbus PDU	Interpretación
0x00	led apagados
0x01	1er led encendido
0x03	1er y 2o led encendidos
0x07	1er, 2o y 3er led encendidos

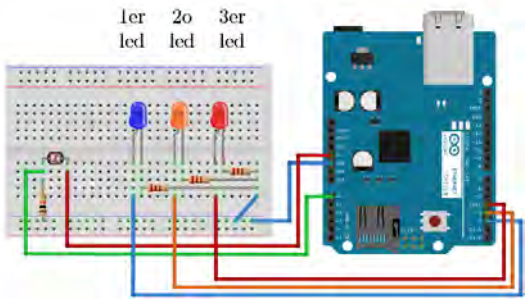


Figura 3-9: Datos en Modbus PDU para el PLC del fotosensor

[3]Led, del inglés *light-emmiting diode* o diodo emisor de luz, es un componente electrónico que emite luz.

Entonces el maestro deberá enviar un paquete *Request* al esclavo como el que se muestra a continuación. Los datos marcados con color verde corresponden al protocolo IP, con color azul los que corresponden al protocolo TCP, con color morado los que corresponden a la cabecera Modbus y con color amarillo los que corresponden al PDU de Modbus.

```
18:14:33.107867 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [P.], seq 2835:2849,
ack 4034, win 29200, length 14
        0x0000: 4500 0036 afb2 4000 4006 0702 c0a8 0145 E..6..@.@......E
        0x0010: c0a8 0178 9c85 01f6 164e 2b0f f4d6 5793 ...x.....N+...W.
        0x0020: 5018 7210 8436 0000 0149 0000 0008 020f P.r..6...I......
        0x0030: 0000 0008 0103                          ......
```

Cabecera de Modbus (MBAP Header):
- Transaction identifier: 0x0149
- Protocol Identifier: 0x0000
- Length: 0x0008
- Unit Identifier: 0x02

Modbus PDU:
- Function Code: 0x0f, Write Multiple Coils (15)
- Starting Address: 0x0000
- Quantity of outputs: 0x0008
- Byte Count: 0x01
- Outputs value: 0x03

En este paquete el maestro envía un *Request* con un identificador de transacción o *Transaction identifier* igual a 0x0149 o 329 en decimal, por lo que se espera que en el paquete *Response* el identificador de transacción corresponda al recibido, es decir al 329. El identificador de protocolo o *Protocol identifier*, corresponde a 0x0000, el valor para Modbus. El valor para el campo *Length* corresponde a 8 bytes, que es el número de bytes que restan, incluyendo el identificador de unidad y la PDU. Para este caso el maestro a asignado como identificador de unidad o *Unit identifier* al esclavo con el valor de 2.

El código de función es el 15, el cual corresponde a la función *Write Multiple Coils*[4], esto significa que el maestro le manda una instrucción al esclavo de escritura a la memoria. El acceso a la memoria del esclavo será desde el inicio, lo indica con el valor 0x0000 del campo *Starting Address*. La cantidad de salidas o *Quantity of outputs* será de 8 bits, que corresponden a los tres bits para los led más 5 bits de relleno. La cantidad de bytes o *Byte Count* a usar está en función de la cantidad de salidas y se calcula como

$$N = \textit{Cantidad de salidas}/8,$$

para este caso

$$N = 8/8 = 1 \text{ byte}.$$

Los valores de salida o *Outputs value* corresponden a la forma en cómo se requieren encender los led, como ya se explicó antes, para este caso su valor es 0x03. Con estos valores es como el maestro construye el PDU del paquete *Request*.

Después del paquete *Request* por parte del maestro, el esclavo envía un paquete *Response* en respuesta al primero, a continuación se muestra el paquete *Response* del esclavo.

[4]En el protocolo Modbus *Coil* hace referencia a un valor binario o discreto, a una salida discreta, por ejemplo encendido o apagado (*on* u *off*).

```
18:14:33.129699 IP 192.168.1.120.502 >192.168.1.69.40069: Flags [P.], seq 4034:4046,
ack 2849, win 2048, length 12
       0x0000: 4500 0034 0f9c 4000 8006 671a c0a8 0178 E..4..@...g....x
       0x0010: c0a8 0145 01f6 9c85 f4d6 5793 164e 2b1d ...E......W..N+.
       0x0020: 5018 0800 f3fb 0000 0149 0000 0006 020f P........I......
       0x0030: 0000 0008                               ....
```

Cabecera de Modbus (MBAP Header):
- Transaction identifier: 0x0149
- Protocol Identifier: 0x0000
- Length: 0x0006
- Unit Identifier: 0x02

Modbus PDU:
- Function Code: 0x0f, Write Multiple Coils (15)
- Starting Address: 0x0000
- Quantity of outputs: 0x0008

El identificador de transacción corresponderá al mismo que el enviado por el maestro, de esta forma el protocolo Modbus establece un control entre solicitudes y respuestas. El identificador de protocolo corresponde al 0. En este caso el paquete *Response* en más pequeño que el *Request*, porque el valor del campo *Length* es de 6 bytes, este valor corresponde a los bytes que restan en el paquete. El identificador de unidad debe permanecer igual que el enviado en el *Request*.

El código de función debe ser el mismo que en el *Request*, que corresponde a la función *Write multiple coils*. El valor inicial de memoria es 0x0000 que fue al que se accedió y corresponde al mismo que en el *Request* del maestro. El valor de salidas corresponde al número que se escribieron en la memoria del esclavo, que corresponde a 8 bits, 3 para los led más 5 para completar un byte. En este mismo paquete se envía adicionalmente de la bandera PUSH de TCP, la bandera ACK para indicar que el paquete anterior de TCP fue recibido satisfactoriamente, es decir, el enviado por el maestro.

En el siguiente paquete se observa la confirmación por parte del maestro de la llegada del paquete *Response* del esclavo, de esto se encarga el protocolo TCP al enviar un paquete con la bandera ACK activada y el número de confirmación esperado.

```
18:14:33.129964 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [.], ack 4046, win 29200,
length 0
       0x0000: 4500 0028 afb3 4000 4006 070f c0a8 0145 E..(..@.@......E
       0x0010: c0a8 0178 9c85 01f6 164e 2b1d f4d6 579f ...x.....N+...W.
       0x0020: 5010 7210 8428 0000                     P.r..(..
```

De esta forma en que se establece el intercambio de información entre el maestro y el esclavo utilizando el protocolo Modbus para TCP/IP, en [Modbus Organization, 2006] se recomienda, para las transacciones entre el maestro y el esclavo, mantener la conexión TCP abierta y no estar abriendo y cerrando ésta por cada transacción.

La interpretación para el PLC del cual se obtuvo el tráfico de red mostrado, corresponde a los dispositivos maestro y esclavos utilizados en este trabajo, en el Capítulo 6 se detalla la infraestructura utilizada. Es importante señalar que el significado físico de los datos dependerá de la infraestructura utilizada. En el Apéndice B se presentan otros ejemplos de transacciones Modbus obtenidos de los dispositivos usados.

3.9. Recomendaciones de implementación para Modbus TCP/IP

Es necesario para la comunicación entre los dispositivos esclavos y maestros que sea establecida una comunicación entre ellos usando el protocolo TCP. En [Modbus Organization, 2006] se dan las consideraciones para la implementación de las conexiones TCP, algunas de las cuales se explican a continuación.

- El número de conexiones entre los clientes y los servidores dependerá de las capacidades de cómputo del dispositivo. En caso de que las conexiones excedan el número autorizado, se recomienda que se cierren las que estén sin usarse, empezando por las que llevan más tiempo en este estado.
- Una vez que la conexión TCP ha sido establecida entre un cliente y un servidor, se recomienda mantenerla abierta para todas las transacciones Modbus entre ambos, en lugar de abrir y cerrar una conexión por cada transacción. El cliente deberá ser capaz de cerrar la conexión a solicitud del servidor, la conexión podrá ser reabierta cuando sea requerida.
- Varias transacciones Modbus pueden llevarse a cabo de forma simultánea en la misma conexión TCP, para esto es necesario que el identificador de la transacción Modbus sea utilizado para identificar unívocamente las solicitudes y las respuestas.
- Se recomienda para un cliente establecer un mínimo de conexiones TCP con un mismo servidor, por ejemplo una conexión por aplicación.
- En caso de que se llevo a cabo una comunicación bidireccional entre dos dispositivos Modbus, es necesario abrir conexiones separadas para el flujo de datos del cliente y para el flujo de datos del servidor.
- Un paquete TCP sólo puede contener un mensaje Modbus, es decir, sólo puede contener un ADU de Modbus. Se recomienda no enviar múltiples solicitudes (*Request*) o respuestas (*Response*) de Modbus en el mismo paquete TCP, es decir no se deben enviar en el mismo PDU de TCP. Aunque esto se recomienda, se observa en dos capturas de tráfico de ICS disponibles en el repositorio CloudShark del año 2012 [Ask Wireshark, 2012] y [Cloudshark, 2012], que el maestro suele enviar en un mismo mensaje TCP varias solicitudes, cada una con su propio Identificador de transacción, de la misma forma los esclavos suelen enviar en un mismo paquete TCP varias respuestas, a continuación se muestra este comportamiento.

  En la siguiente captura de tráfico se observa cómo el maestro (dispositivo con dirección IP 141.81.0.10) envía cinco solicitudes de Modbus diferentes al esclavo (dispositivo con dirección IP 141.81.0.26), con Identificadores de transacción 0x4877, 0x4878, 0x4879, 0x487a y 0x487b respectivamente.

```
06:03:04.506835 IP 141.81.0.10.51411 >141.81.0.26.502: Flags [P.],
seq 165112417:165112477, ack 890920318, win 63725, length 60
       0x0000: 4500 0064 7581 4000 8006 0000 8d51 000a E..du.@......Q..
       0x0010: 8d51 001a c8d3 01f6 09d7 6a61 351a 5d7e .Q........ja5.]
       0x0020: 5018 f8ed 1b1d 0000 4877 0000 0006 ff04 P.......Hw......
       0x0030: 0001 0063 4878 0000 0006 ff04 0029 0002 ...cHx.......)..
       0x0040: 4879 0000 0006 ff04 08ab 0016 487a 0000 Hy.........Hz..
       0x0050: 0006 ff04 08d2 0002 487b 0000 0006 ff02 ........H{......
       0x0060: 0063 001e                               .c..
```

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x4877
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Starting Address: 0x0001
Quantity of Input Registers: 0x0063

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x4878
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Starting Address: 0x0029
Quantity of Input Registers: 0x0002

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x4879
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Starting Address: 0x08ab
Quantity of Input Registers: 0x0016

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x487a
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Starting Address: 0x08d2
Quantity of Input Registers: 0x0002

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x487b
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x02, Read Discrete Inputs
Starting Address: 0x0063
Quantity of Input Registers: 0x001e

Entonces el esclavo envía un paquete TCP con cinco Respuestas de Modbus para las solicitudes del maestro, cuyo Identificadores de transacción son 0x4877, 0x4878, 0x4879, 0x487a y 0x487b respectivamente, es decir se llevaron a cabo cinco transacciones Modbus.

```
06:03:04.555282 IP 141.81.0.26.502 >141.81.0.10.51411: Flags [P.],
seq 890920318:890920617, ack 165112477, win 600, length 299
       0x0000: 4500 0153 b27e 0000 4006 ac60 8d51 001a E..S. ..@..‘.Q..
       0x0010: 8d51 000a 01f6 c8d3 351a 5d7e 09d7 6a9d .Q......5.] ..j.
       0x0020: 5018 0258 629e 0000 4877 0000 00c9 ff04 P..Xb...Hw......
       0x0030: c600 3200 0030 5830 3036 3035 3233 3638 ..2..0X006052368
       0x0040: 3500 0000 0000 0100 0030 3030 3030 3030 5........0000000
       0x0050: 3030 3030 3033 3033 3330 3700 0000 0000 00000303307.....
       0x0060: 0000 0000 0000 0000 0000 0400 0000 0000 ................
       0x0070: 0000 0000 0000 0000 0000 0000 0000 0100 ................
       0x0080: 0000 0400 0000 0000 0000 0000 0000 0000 ................
       0x0090: 0010 bc00 0010 c400 004e b000 0100 0300 .........N......
       0x00a0: 0000 3c00 0000 0000 0000 0000 0000 0000 ..<.............
       0x00b0: 0000 0000 0000 0000 0070 6f61 7000 6300 .........poap.c.
       0x00c0: 0000 0000 0000 0000 0000 0000 0070 6f61 .............poa
       0x00d0: 7000 6300 0000 0000 0000 0000 0000 0000 p.c.............
       0x00e0: 0070 0061 7000 6300 0000 0000 0000 0000 .p.ap.c.........
       0x00f0: 0000 0000 0000 1448 7800 0000 07ff 0404 .......Hx.......
       0x0100: 0004 0000 4879 0000 002f ff04 2c00 0000 ....Hy.../..,...
```

```
0x0110: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0x0120: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0x0130: 0000 0000 0000 0000 0048 7a00 0000 07ff .........Hz.....
0x0140: 0404 0000 0000 487b 0000 0007 ff02 04bd ......H{........
0x0150: 4f67 09                                 Og.
```

Cabecera de Modbus (MBAP Header):
Identificador de transacción: 0x4877
Protocol Identifier: 0x0000
Length: 0x00c9
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Byte Count: 0xc6
Register 2258 (UINT16): 0x0032
Register 2259 (UINT16): 0x0000
...
Register 2355 (UINT16): 0x0000
Register 2356 (UINT16): 0x0014

Cabecera de Modbus (MBAP Header):
Identificador de transacción: 0x4878
Protocol Identifier: 0x0000
Length: 0x0007
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Byte Count: 0x04
Register 2258 (UINT16): 0x0004
Register 2259 (UINT16): 0x0000

Cabecera de Modbus (MBAP Header):
Identificador de transacción: 0x4879
Protocol Identifier: 0x0000
Length: 0x002f
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Byte Count: 0x2c
Register 2258 (UINT16): 0x0000
Register 2259 (UINT16): 0x0000
...
Register 2278 (UINT16): 0x0000
Register 2279 (UINT16): 0x0000

Cabecera de Modbus (MBAP Header):
Identificador de transacción: 0x487a
Protocol Identifier: 0x0000
Length: 0x0007
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x04, Read Input Registers
Byte Count: 0x04
Register 2258 (UINT16): 0x0000
Register 2259 (UINT16): 0x0000

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x487b
Protocol Identifier: 0x0000
Length: 0x0007
Unit Identifier: 0xff

Modbus PDU:
Function Code: 0x02, Read Discrete Inputs
Byte count: 0x04
Input Status: 0xbd4f6709

De acuerdo al análisis realizado a esta captura se determinó que estos paquetes mostrados como ejemplo corresponden a un dispositivo esclavo Elau PLC, ahora bajo la marca Schneider Electric. De esta forma se demuestra que en esta implementación del protocolo Modbus TCP/IP no se siguió la recomendación de la especificación, pero pareciera ser que esta forma de implementarlo, de enviar en un mismo paquete TCP varios mensajes de Modbus, es común en los dispositivos de ICS.

Si se desea profundizar aún más sobre el protocolo Modbus, por ejemplo sobre el resto de las funciones o sobre los códigos de excepción, se recomienda consultar [Organization, 2012]; si se desea conocer sobre Modbus para TCP, por ejemplo sobre como implementarlo a detalle o ejemplos de como deben operar las funciones, se recomienda consultar [Modbus Organization, 2006] y [Modbus Organization, 2009]; o sobre Modbus Serial, consultar [Modbus Organization, 2002].

Capítulo 4

Seguridad en el protocolo Modbus TCP/IP

4.1. Resumen

Los ICS tienen una infraestructura crítica porque son sistemas de industria petrolera, hidráulica, eléctrica, entre otros, por lo que una brecha de seguridad en ellos podría llegar a tener un impacto de gran consideración, como poner en riesgo la vida de las personas. El protocolo Modbus TCP/IP es ampliamente utilizado en los ICS, es por eso necesario su estudio desde el punto de vista de la seguridad para detectar vulnerabilidades y analizar que riesgos se pueden presentar, en este capítulo se revisan la taxonomía de ataques hacia este protocolo, esto permite tener una idea clara de la naturelza y alcance de los ataques a Modbus TCP/IP, esta revisión se toma principalmente de [Huitsing et al., 2008].

4.2. Seguridad de la información.

La seguridad de la información es el conjunto de procedimientos, mecanismos y medidas preventivas y reactivas que permiten garantizar la confidencialidad, integridad y disponibilidad de la información, es decir, que permiten resguardar y proteger la información. El campo de la seguridad de la información abarca tanto sistemas físicos como sistemas lógicos. Los conceptos de confidencialidad, integridad y disponibilidad, que son conocidos como Pilares de la seguridad de la información [López Barrientos and Quezada Reyes, 2006], se definen a continuación.

- La confidencialidad es la capacidad para que la información, pueda ser accedida e interpretada únicamente por las personas autorizadas para ello.
- La integridad busca prevenir que cualquier modificación no autorizada sobre la información sea detectada.
- La disponibilidad se cumple si la información puede ser accedida siempre que se requiera por usuarios autorizados.

Otros elementos a considerar dentro de la seguridad de la información son los siguientes:

- El no repudio busca ser una garantía de que en una comunicación, tanto el remitente del mensaje no pueda negar el haberlo enviado, como el destinatario no pueda negar el haberlo recibido.
- La autenticación es el proceso de verificar la identidad, es decir, la verificación de que el usuario es quien dice ser. Se realiza a través de examinar las credenciales proporcionadas, éstas pueden ser algo que se sabe, algo que se tiene o algo que se es.
- El control de acceso consiste en la capacidad de otorgar o negar el acceso a la información, comúnmente después de un proceso de autenticación.

Es necesario también definir tres elementos que pueden llegar a afectar a la información, estos son:

- La amenaza es todo aquello que intente producir algún daño a la información.
- La vulnerabilidad es una debilidad en un sistema, persona, etc, que puede ser explotada para afectar a alguno de los Pilares de la información y por lo tanto a la información.
- El riesgo es la posibilidad de que un elemento humano, natural o informático atente contra la información. El riesgo se incrementa conforme aumente el número de amenazas y vulnerabilidades de la información.

Un proceso fundamental de la seguridad de la información es el análisis o evaluación de riesgos, el cual es un proceso iterativo que consiste en identificar la probabilidad de que un evento no deseado afecte a la información, y el impacto que tendrá. Con este análisis se podrá decidir si el riesgo se mitiga, transmite o acepta. Como parte del análisis de riesgos es importante conocer la vulnerabilidades, y amenazas que atenten contra la confidencialidad, integridad y disponibilidad, es por eso que en este trabajo se trata la seguridad y los tipos de ataques al protocolo Modbus.

4.3. Problemas de seguridad en el protocolo Modbus

Cuando se creó el protocolo Modbus en la década de 1970 no se consideró la importancia de contar con mecanismos de seguridad que se incluyeran en la especificación de este protocolo. El protocolo no tiene forma de cifrar los datos, de revisar la integridad de los mensajes, o de establecer una autenticación entre el maestro y el esclavo [Jiménez Díaz, 2011].

No existe algún protocolo adicional en Modbus para el cifrado de los datos, éstos se transmiten en claro, por lo que si son interceptados pueden ser fácilmente falsificados, además de que pueden revelar información sobre la configuración y uso del dispositivo de control [McLaughlin et al., 2016], por lo tanto revelar información sobre el proceso industrial.

De acuerdo a la especificación descrita en [Modbus Organization, 2002], la PDU de Modbus serial, contiene el campo *Error Checking*, de 2 bytes, y es utilizado para comprobar si han ocurrido errores durante la transmisión de los datos, es decir, es un código detector de errores; en la especificación para TCP/IP, a diferencia de Modbus Serial, no se revisa la

integridad del PDU porque no existe un campo similar al de Modbus Serial, es decir, cuando el receptor recibe la PDU de Modbus TCP/IP no puede darse cuenta si han ocurrido errores en la transmisión, se puede considerar que Modbus TCP/IP deja este verificación de errores durante la transmisión de datos al protocolo TCP por que cuenta con un campo para la verificación de errores durante la transmisión, este campo se describe en [RFC793, 1981]; es importante señalar que la verificación de errores durante la transmisión es diferente a verificar la integridad de los datos de Modbus enviados, por que el código que utiliza TCP para detectar los errores de transmisión consiste en aplicar una operación XOR entre grupos de 16 bits, es posible que aunque existan cambios a nivel de bits durante la transmisión el resultado de la operación XOR sea el mismo, esto por su naturaleza, por lo que es posible que no se detecten errores de transmisión y aún así puedan existir cambios en los datos de Modbus.

No existe una autenticación entre el maestro y el esclavo, es decir, no se establece una relación de identificación de los dispositivos que se van a comunicar para enviar las peticiones y respuestas del protocolo Modbus, entonces, cualquier dispositivo no autorizado podría recibir o enviar paquetes Modbus a otro dispositivo, en el PDU de Modbus solamente se cuenta con el campo de Identificador de transacción o *Transaction Identifier*, el cual es descrito en [Modbus Organization, 2006] y explicado en el capítulo 3 de este trabajo, pero este identificador solamente tiene como función establecer una relación entre un paquete de solicitud o *Request* y su respuesta o *Response*, permite llevar un control que asocie a cada solicitud con su respuesta. Esto podría facilitar ataques de hombre en medio (Man in the middle, MiM), o ataques de repetición o *replay* [McLaughlin et al., 2016], en el cual después de aplicar un ataque de hombre en medio, se reenvían los datos al destino.

Los efectos de los ataques a sistemas que ocupan el protocolo Modbus pueden ir desde la interrupción esporádica del funcionamiento de los dispositivos esclavos, o la deshabilitación del maestro, hasta la pérdida del control de estos dispositivos esclavo por suplantación del dispositivo maestro, y/o obtener el control total del maestro. De acuerdo a [Huitsing et al., 2008] los ataques a sistemas y redes que usen el protocolo Modbus pueden agruparse en tres categorías,

- Ataques que exploten vulnerabilidades en la especificación del protocolo Modbus. Este tipo de ataques son comunes a todos los sistemas y redes que sigan la implementación del protocolo Modbus definida en [Organization, 2012], [Modbus Organization, 2009] y [Modbus Organization, 2006]. Es posible que algunos fabricantes no incluyan todas las funciones de la implementación, por ejemplo las funciones de diagnóstico. El impacto de un ataque de esta categoría afectará a una cantidad considerable de dispositivos.

- Ataques que exploten vulnerabilidades de implementaciones específicas del protocolo Modbus realizada por algún fabricante. A diferencia de la categoría anterior, un ataque de este tipo afectaría al menos a los dispositivos para los que se diseñó dicho ataque.

- Ataques a la infraestructura. Son ataques relacionados con dispositivos de TI, de redes y de telecomunicaciones que conformen la infraestructura de ICS.

Este trabajo contempla solamente el estudio de ataques a la primera categoría, es decir a las vulnerabilidades de la especificación del protocolo Modbus.

En [Zhu et al., 2011] se explica que existen diferentes formas de explotar las vulnerabilidades del protocolo Modbus usando los códigos de función, por ejemplo a través de los códigos de función 0x05, 0x0F, 0x06 y 0x10, mostrados en la tabla 3.3 que se usan para escritura en el dispositivo esclavo, debido a que el protocolo Modbus no cuenta con un mecanismo de autenticación, un dispositivo esclavo no podrá diferenciar entre la información de escritura a sus registros enviada por un dispositivo maestro legítimo a la enviada por un maestro malicioso. Esto se puede lograr al secuestrar una conexión TCP existente entre un dispositivo esclavo y un maestro e inyectar paquetes suplantados dentro del flujo TCP, reiniciar una conexión existente y crear una nueva conexión [Goldenberg and Wool, 2013].

4.4. Taxonomía de ataques al protocolo Modbus

En [Huitsing et al., 2008] se describen los ataques al protocolo Modbus divididos en tres categorías para su mejor estudio,

1. ataques que afectan solamente a Modbus Serial,
2. ataques que afectan tanto a Modbus Serial como a Modbus TCP,
3. ataques que afectan solamente a Modbus TCP.

Aquí se describirán únicamente los ataques de la segunda y tercera categoría. Los ataques han sido propuestos por [Huitsing et al., 2008], y uno más descrito en [Bhatia et al., 2014].

Para llevar a cabo estos ataques se requiere el uso de un analizador de protocolos (comúnmente conocido como *sniffer*) y un generador de paquetes de Modbus, junto con acceso a la red en la que se encuentre el maestro y el esclavo.

- Ataques a la confidencialidad. Este tipo de ataques se lleva a cabo a través de la obtención de información de los paquetes Modbus, ya sea tanto del maestro como de las configuraciones del esclavo, en el caso de éste último configuraciones o datos de los dispositivos de control a los que está conectado; obtención de información de la red como configuraciones de la red, etc.

- Ataques a la integridad. Se refiere a la modificación de paquetes Modbus, modificar la configuración de a quien se debe enviar o recibir los paquetes Modbus, transmitir datos con información maliciosa al maestro (datos con información errónea que se cree vienen del esclavo), o enviar mensajes Modbus maliciosos hacia el esclavo para reconfigurarlo. Es posible realizarlo al suplantar al maestro o al esclavo.

- Ataques a la disponibilidad. Resultan en una denegación del servicio ya sea al maestro, al esclavo, o al acceso a la red. Particularmente los esclavos pueden perder funcionalidad, reiniciarse o colapsarse. Es posible bloquear paquetes Modbus y deshabilitar la comunicación en la red.

A continuación se detallan los ataques al protocolo Modbus, primero los ataques comunes a Modbus Serial y TCP/IP, y después los ataques para Modbus TCP/IP.

4.4.1. Ataques a Modbus Serial y TCP/IP

- *Broadcast message spoofing.* Suplantación de mensajes de difusión o *broadcast*, este ataque consiste en el envío de mensajes falsos de tipo *broadcast* a los esclavos. Es difícil de detectar por que una vez que el esclavo recibe el mensaje falso, no envía un mensaje de respuesta al maestro. [Huitsing et al., 2008]

- *Baseline response replay.* Se requiere interceptar y capturar tráfico normal entre un maestro y un esclavo, este tráfico será la línea base de la cual se reenviarán algunos de los mensajes capturados devuelta al maestro. [Huitsing et al., 2008]

- *Direct slave control.* Control directo del esclavo, para realizar este ataque es necesario bloquear al maestro y controlar uno o más dispositivos esclavos. Este es uno de los ataques de mayor impacto. [Huitsing et al., 2008]

- *Modbus network scanning.* Este ataque consiste en el envío de mensajes no anómalos a todas las posibles direcciones de dispositivos en una red Modbus, para obtener información sobre éstos. Por ejemplo, en Modbus TCP/IP se enviarían mensajes a todas las direcciones IP del segmento de la red a la cual están conectados los dispositivos. [Huitsing et al., 2008]

- *Passive reconnaissance.* Reconocimiento pasivo, consiste en capturar de forma pasiva mensajes Modbus o en capturar tráfico de esa red. [Huitsing et al., 2008]

- *Response delay.* Se retrasan los mensajes de respuesta o *Response* para que el maestro reciba información no actualizada de los dispositivos esclavo. [Huitsing et al., 2008]

- *Rogue interloper.* Este ataque consiste en colocar en una computadora un dispositivo no autorizado, ya sea serial o Ethernet, el cual se conectará a una red no segura distinta a la de la infraestructura de ICS. Este dispositivo podrá leer, modificar o fabricar mensajes Modbus o tráfico de red. Este tipo de ataque es de un mayor impacto. [Huitsing et al., 2008]

4.4.2. Ataques a Modbus TCP/IP

- *Irregular TCP framing.* De acuerdo a [Modbus Organization, 2006] se debe de enviar un sólo paquete TCP con un sólo Modbus ADU, es decir, no se recomienda que múltiples mensajes Modbus se envíen en un paquete TCP. Este ataque inyecta paquetes TCP formados con múltiples mensajes Modbus o modifica mensajes legítimos para crear paquetes TCP con múltiples mensajes Modbus, con lo cual se puede causar el cierre de una conexión de un maestro o de un esclavo [Huitsing et al., 2008]. Aunque como se explicó en el capítulo 3, no todos los dispositivos implementan esta característica.

- *TCP FIN Flood.* Inundación de paquetes TCP con la bandera FIN activa, este ataque envía paquetes TCP con la bandera de FIN activada, suplantando a uno de los dispositivos que intervienen en la comunicación, después de un mensaje Modbus legítimo entre un maestro y un esclavo, para cerrar la conexión TCP. [Huitsing et al., 2008]

- *TCP pool exhaustion.* La especificación de Modbus TCP/IP [Modbus Organization, 2006] describe dos clases de grupos (*pools*) de conexión, un grupo de conexión con prioridad y uno sin prioridad. Una vez agotadas las conexiones en estos grupos, un

dispositivo Modbus no podrá aceptar nuevas conexiones. Para realizar el ataque es necesario establecer un gran número de conexiones TCP maliciosas con un dispositivo Modbus, usando direcciones IP marcadas para conexiones con prioridad, y direcciones IP no marcadas para conexiones no prioritarias. La actividad de la red debe ser mantenida en todas estas conexiones maliciosas para lograr un ataque de denegación de servicio, por que el dispositivo podría cerrar las conexiones maliciosas al estar sin uso [Modbus Organization, 2006], de esta forma las conexiones maliciosas saturan al dispositivo, por lo tanto las conexiones legítimas no podrían ser atendidas. [Huitsing et al., 2008].

- *TCP RST Flood*. Inundación de paquetes TCP con la bandera RST activa, este ataque envía paquetes TCP con la bandera de RST activada, suplantando a uno de los dispositivos que intervienen en la comunicación, después de un mensaje Modbus legítimo entre un maestro y un esclavo para cerrar la conexión TCP. [Huitsing et al., 2008]

- *Flooding attack*. Consiste en inyectar una gran cantidad de paquetes hacia los dispositivos esclavos con códigos de función específicos. El objetivo no es evitar que los mensajes del maestro lleguen a los esclavos, sino en controlar los esclavos al enviarles una gran cantidad de paquetes maliciosos, de esta forma los mensajes legítimos del maestro son "ahogados". [Bhatia et al., 2014]

Capítulo 5

Técnicas de detección de intrusos

5.1. Resumen

En gran parte de las redes de computadoras existe tráfico anómalo, es decir, tráfico de red ocasionado ya sea por las malas configuraciones de dispositivos de cómputo o de red, o ya sea por actividad maliciosa asociada a amenazas y vulnerabilidades. Existe software y hardware diseñado para contener las afectaciones provocadas por la actividad maliciosa tales como Firewall, UTM, Next Generation Firewall, IDS, IPS. La mayoría de estos dispositivos usa métodos de detección de intrusos basados en firmas. Además de los métodos basados en firmas existe otra metodología que puede aplicarse a la detección de intrusos que se conoce como detección de anomalías, habiendo varios métodos para llevarla a cabo. En este capítulo se revisa al clasificador ingenuo de Bayes como método de detección de anomalías. Además se estudian las métricas más utilizadas para evaluar la efectividad de un método de detección.

5.2. Basadas en firmas

Los métodos de detección de intrusos basados en firmas consisten en la creación de patrones, básicos o avanzados, para detectar tráfico anómalo, particularmente malicioso, cuando el tráfico analizado coincide con el patrón se activa una alerta para tomar algún tipo de acción, por ejemplo bloquear ese tráfico o solamente alertar. En la figura 5-1 se muestra una firma del IDS Snort para el protocolo Modbus, publicada por Digital Bond bajo el proyecto QuickDraw [Digital Bond, 2011], se observa en ésta que se emitirá una alerta para las conexiones establecidas del protocolo Modbus TCP en donde el tamaño de los datos de la capa de aplicación por paquete sea mayor a 300 bytes, ya sea que los paquetes se envíen desde el maestro (o cliente) hacia el servidor (o esclavo), o en sentido inverso. Esta comprobación se realiza para registrar paquetes de un tamaño anormal para el protocolo Modbus TCP y que puedan ser causa de un ataque de Denegación de Servicio (DoS).

```
alert tcp $MODBUS_CLIENT any <> $MODBUS_SERVER 502 (flow:established; dsize:>300;
msg:"SCADA_IDS: Modbus TCP - Illegal Packet Size, Possible DOS Attack"; reference:
url,digitalbond.com/tools/quickdraw/modbus-tcp-rules; classtype:non-standard-proto
col; sid:1111008; rev:1; priority:1;)
```

Figura 5-1: Firma del IDS Snort

Es necesario crear una lista de firmas adecuada para los servicios de red que se ofrecen en el sistema que se esté analizando, esta lista es la que determinará que tipo de anomalías pueden ser detectadas, así como el grado y el alcance de los eventos que se generarán. Los datos arrojados por las firmas cuando alertan sobre el tráfico anómalo también permiten realizar un análisis sobre otros indicadores como el comportamiento de la red, los datos de la capa de aplicación y la cantidad y tipo de anomalías, ya sean amenazas o vulnerabilidades, detectadas.

Para aumentar la precisión en la detección, es necesario crear la regla lo más específico posible, es decir, adecuar la regla para un ataque o amenaza en particular. Si una firma es configurada sin mucha especificidad, es posible que tráfico normal sea considerado como tráfico anómalo, a esto se le conoce como Falso Positivo (*False Positive*, FP). Para crear firmas que minimicen las alertas de Falsos Positivos y maximicen la detección del tráfico anómalo, conocido como Verdaderos Positivos (*True Positive*, TP), deben ser creadas por personal con altos conocimientos en el área tanto de redes y seguridad, como del protocolo de la capa de aplicación que se esté empleando, en el caso de este trabajo del protocolo Modbus.

Con este tipo de detección de intrusos, solamente se alertará si el tráfico anómalo o normal coincide con el patrón de la firma exactamente. Este método está limitado frente a la aparición de nuevos vectores de ataque que no han sido detectados, por lo tanto no han sido modelados y no se ha creado una firma que pueda detectarlos. Requiere de actualizaciones constantes por la aparición diaria de nuevos tipos de amenazas y vulnerabilidades.

5.3. Detección de anomalías

Los métodos de detección de anomalías están basados en la caracterización del comportamiento de una red. Se modela el comportamiento del tráfico identificando lo que es normal, y tomando en cuenta el comportamiento del tráfico anómalo si es que se cuentan con esos datos, si se detecta algún comportamiento que se sale de este modelo es posible advertir bajo ciertos criterios, que pueda tratarse de tráfico malicioso. En [García-Teodoro et al., 2009] se describen los siguientes tipos de métodos de detección de anomalías.

- Basado en estadísticas. En las técnicas basadas en estadísticas, la actividad de tráfico de red es capturada y es creado un perfil que represente su comportamiento estocástico. Este perfil se basa en métricas tales como la tasa de tráfico, el número de paquetes por protocolo, la tasa de conexiones, el número de direcciones IP, entre otros.

  Son considerados dos conjuntos de datos del tráfico de red durante el proceso de detección de anomalías, una corresponde al perfil del tráfico actual observado en un intervalo de tiempo, y el otro es para el perfil estadístico previamente entrenado. Para la detección, se comparan los dos comportamientos, y la anomalía es determinada con base en esta comparación. Se asigna una puntuación que indicará el grado de irregularidad para un evento específico, entonces se alertará la ocurrencia de una anomalía cuando la puntuación supera un cierto umbral.

- Base de conocimiento. Es considerado como un sistema experto. Los sistemas expertos están destinados a clasificar los datos auditados de acuerdo a una serie de reglas, que involucra tres pasos. Primero, los diferentes atributos y clases son identificados a partir de los datos de entrenamiento. Segundo, un conjunto de reglas de clasificación, parámetros o procedimientos son identificados. Tercero, los datos auditados son clasificados de acuerdo a esto.

- Aprendizaje automatizado. Este tipo de técnicas establecen un modelo explícito o implícito que permita el análisis de patrones para ser categorizados. Una característica de esta técnica, es la necesidad de etiquetar los datos para entrenar el comportamiento del modelo, un procedimiento que demanda grandes cantidades de recursos. Es posible que el modelo cambie su estrategia de ejecución, al adquirir nueva información.

5.3.1. Clasificador ingenuo de Bayes

Una Red Bayesiana es un modelo que permite establecer reglas a partir de las relaciones probabilísticas existentes entre las variables en un problema. Estas redes permiten representar conocimiento y razonamiento en problemas en donde se presenta incertidumbre. Una Red Bayesiana $B = (N, A, \Theta)$ es una grafo acíclico dirigido (N, A) en donde cada nodo $n \in N$ representa una variable de un dominio, por ejemplo una serie de atributos, y cada arco $a \in A$ entre los nodos representan un dependencia probabilística entre las variables, la cual se puede cuantificar usando un distribución de probabilidad condicional $\theta_i \in \Theta$ para cada nodo n_i. Una Red Bayesiana se puede utilizar para calcular la probabilidad condicional de un nodo, dados los valores asignados a los otros nodos. Cuando son usadas en conjunto con métodos estadísticos, las Redes Bayesianas tienen varias ventajas para analizar datos. [Bhattacharyya and Kalita, 2014]

Los clasificadores ingenuos de Bayes (NB, por las siglas en inglés de naive Bayes) son Redes Bayesianas sencillas que están formadas por un grafo acíclico dirigido con solamente un nodo raíz llamado nodo padre, este grafo representa un nodo no observado con varios hijos que corresponden a los nodos observados, se asume la independencia entre los nodos hijos en el contexto de sus nodos padres. El modelo ingenuo de Bayes es un modelo probabilístico Bayesiano simplificado. Calcula la probabilidad de que un evento suceda dadas las variables o atributos relacionadas al evento. Se asume que la probabilidad de que una variable, dado que el resultado final ocurre, es independiente de la probabilidad de otras variables dado que ocurren los mismos resultados finales, es decir, las variables no deben ser dependientes entre sí. Cuando se observa un conjunto de clases en los datos de entrenamiento, el clasificador ingenuo de Bayes asigna los datos observados a la clase con la probabilidad más alta. Durante el entrenamiento, el algoritmo ingenuo de Bayes calcula las probabilidades de un resultado dado una variable particular y luego almacena esta probabilidad. Esto se repite para cada variable. [Bhattacharyya and Kalita, 2014]

La clasificación se realiza ocultando al nodo padre, es decir, se desconoce la clase a la que pertenece el nodo padre, se calcula a qué clase debe pertenecer el registro u objeto del conjunto de pruebas, cada registro del conjunto de pruebas está conformado por las variables, las cuales son representadas por los nodos hijos. En el conjunto de datos de entrenamiento

sólo se deben de calcular las probabilidades condicionales, porque la estructura es única. Una vez que se ha calculado la red, es posible clasificar cualquier objeto nuevo a través de sus variables usando la regla de Bayes. Dado que el modelo ingenuo de Bayes opera bajo el supuesto de que las variables del objeto son independientes, su probabilidad combinada se obtiene de la siguiente forma:

$$P(c_i \mid A) = \frac{P(a_i \mid c_i)P(a_2 \mid c_i)...P(a_n \mid c_n)}{P(A)}$$

en donde P(A) es determinada por la condición de normalización. En la fase de prueba, el tiempo que toma calcular la probabilidad de la clase dada por cada registro es en el peor de los casos, proporcional al número de variables. [Bhattacharyya and Kalita, 2014]

De acuerdo a [Bhattacharyya and Kalita, 2014] las limitaciones de este método son:

- La capacidad de clasificación del modelo ingenuo de Bayes es idéntica a un sistema basado en umbrales que obtiene la suma de las salidas obtenidas de los nodos hijos.
- Dado que los nodos hijos no interactúan entre sí y sus resultados sólo influyen en la probabilidad del nodo raíz, es difícil incorporar información adicional, ya que no hay interacción directa entre las variables que contienen información y los nodos hijos.

Para aplicar este método en la detección de anomalías, se obtienen datos del tráfico de red del sistema, estos datos servirán para encontrar las probabilidades para determinar si un registro u objeto, por ejemplo una conexión, pertenece a la clase del tráfico con anomalías o a la no clase, es decir al tráfico normal. Cuando nuevo tráfico de red llega, el clasificador utilizará el teorema de Bayes para decidir a qué clase pertenece el tráfico.

Se realizan los cálculos de las probabilidades de las variables y sus valores con el fin de determinar su relevancia en la predictibilidad de la pertenencia o no a la clase del tráfico con anomalías. A continuación se enlistan los parámetros y significados utilizados en estos cálculos.

- N número de registros, $N = Nc + N\overline{c}$
- Nc número de registros que pertenecen a la clase de tráfico anómalo
- $N\overline{c}$ número de registros que pertenecen a la no clase del tráfico anómalo, es decir, al tráfico normal
- x_i es un valor que puede tomar una variable o atributo, una variable puede tener tantos valores como sean necesarios, $var_j[x_1, x_2, x_3, ..., x_n]$
- Nx número de registros con la variable o atributo var_j y valor x_i, es decir $var_j[x_i]$
- Ncx número de registros con la variable y valor $var_j[x_i]$, y que pertenecen a la clase de tráfico malicioso
- $N\overline{c}x$ número de registros con la variable y valor $var_j[x_i]$, y que pertenecen a la no clase de tráfico anómalo, es decir, a la clase de tráfico normal

Para el cálculo de probabilidades se realizan los siguientes cálculos.

- $P(c)$ probabilidad de que un registro pertenezca a la clase de tráfico anómalo

$$P(c) = \frac{Nc}{N}$$

- $P(\overline{c})$ probabilidad de que un registro pertenezca a la no clase de tráfico anómalo

$$P(c) = \frac{N\overline{c}}{N}$$

- $P(c \mid x)$ probabilidad de que un registro pertenezca a la clase de tráfico anómalo dado que tenga la variable con el valor $var_j[x_i]$

$$P(c \mid x) = \frac{Ncx}{Nx}$$

- $P(x \mid c)$ probabilidad de que un registro tenga la variable con el valor $var_j[x_i]$ dado que pertenezca a la clase de tráfico anómalo

$$P(x \mid c) = \frac{Ncx}{Nc}$$

- $P(x \mid \overline{c})$ probabilidad de que un registro tenga la variable con el valor $var_j[x_i]$ dado que pertenezca a la no clase de tráfico anómalo

$$P(x \mid \overline{c}) = \frac{N\overline{c}x}{Nc}$$

- *Epsilon* Determina si la variable con el valor $var_j[x_i]$ es relevante en términos de predictibilidad, es decir, a cuántas desviaciones estándar está $P(c \mid x)$

$$Epsilon = \frac{Nx \cdot P(c \mid x) - P(c)}{[Nx \cdot P(c) \cdot (1 - P(c))]^{1/2}}$$

- *Score* Asigna una puntuación o peso a la característica y valor $var_j[x_i]$, para conocer su influencia en la predicción de pertenencia a la clase, de cada registro

$$\ln \frac{P(x \mid c)}{P(x \mid \overline{c})}$$

5.4. Métricas de evaluación

Es necesario contar con métodos de evaluación del modelo de detección de anomalías para conocer su efectividad. El tráfico de red se modela etiquetándolo en dos clases, al tráfico anómalo o anormal como Positivo o *Positive* (P), y al tráfico normal como Negativo o *Negative* (N). Cuando el modelo clasifique correctamente al tráfico de red se usará la etiqueta de Verdadero o *True* (T), cuando la clasificación sea errónea se usará la etiqueta Falso o *False* (F).

Cuando se clasifica correctamente se tienen dos casos, si se clasifica correctamente al tráfico anómalo entonces el resultado es un Verdadero Positivo o *True Positive* (TP), si se clasifica correctamente al tráfico normal el resultado es un Verdadero Negativo o *True Negative* (TN). De forma similar se tienen dos casos cuando se clasifica de forma incorrecta, si al tráfico normal se le clasifica como anómalo el resultado es un Falso Positivo o *False Positive* (FP), mientras que si al tráfico anómalo se le clasifica como normal, es decir que el tráfico anómalo no es detectado, el resultado es un Falso Negativo o *False Negative* (FN) [Bhattacharyya and Kalita, 2014]. En la Matriz de confusión de la figura 5-2 se muestran estos casos.

	Verdadero (True, T)	Falso (False, F)
Positivo (Positive, P)	TP	FP
Negativo (Negative, N)	TN	FN
	Predicción correcta (TP y TN)	Predicción errónea (FP y FN)

Figura 5-2: Matriz de confusión

En este trabajo se usarán las siguientes medidas para medir la efectividad del clasificador, sensibilidad, especificidad, precisión, falsos descubrimientos, exactitud, clasificación errónea, coeficiente de correlación de Mathews,y curvas ROC, con estas medidas es posible conocer las tasas de detección y de fallo del clasificador ingenuo de Bayes. Se explican a continuación con base en [Bhattacharyya and Kalita, 2014] y [Burgueño et al., 1995].

- Sensibilidad (*True Positive Rate*, TPR). Es la razón entre el tráfico anómalo correctamente clasificado (TP) y el tráfico anómalo total, éste último se compone de la suma del tráfico anómalo correctamente clasificado (TP), más el tráfico anómalo erróneamente clasificado (FN). También se le conoce como tasa de aciertos o *hit rate*. Se le da mayor prioridad a la sensibilidad sobre la especificidad cuando se requiere que se detecten todas las anomalías en el sistema, es decir, cuando se quiere proteger el sistema a cualquier costo.

$$TPR = \frac{\text{tráfico anómalo correctamente clasificado}}{\text{tráfico anómalo total}} = \frac{TP}{TP + FN}$$

- Especificidad (*True Negative Rate*, TNR). Es la razón entre el tráfico normal correctamente clasificado (TN) y el tráfico normal total, éste último se compone de la suma del tráfico normal correctamente clasificado (TN), más el tráfico normal erróneamente clasificado (FP). Se la da mayor prioridad a la especificidad cuando se requiere mayor eficiencia en el sistema.

$$TNR = \frac{\text{tráfico normal correctamente clasificado}}{\text{tráfico normal total}} = \frac{TN}{TN + FP}$$

- Precisión (*Positive Predictive Value*, PPV). Es la razón del tráfico anómalo correctamente clasificado (TP), entre el total del tráfico clasificado como anómalo, éste último se compone de la suma del tráfico anómalo correctamente clasificado (TP), más el tráfico normal clasificado como anómalo (FP).

$$PPV = \frac{\text{tráfico anómalo correctamente clasificado}}{\text{total del tráfico clasificado como anómalo}} = \frac{TP}{TP + FP}$$

- Falsos descubrimientos. (*False Discovery Rate*, FDR). Es la razón del tráfico normal clasificado como anómalo (FP), entre el total del tráfico clasificado como anómalo, éste último se compone de la suma del tráfico anómalo correctamente clasificado (TP), más el tráfico normal clasificado como anómalo (FP).

$$FDR = \frac{\text{tráfico normal clasificado como anómalo}}{\text{total del tráfico clasificado como anómalo}} = \frac{FP}{TP + FP}$$

- Exactitud (*Accuracy*, ACC). Es la razón de la suma de las predicciones correctas del tráfico anómalo y normal (TP + TN), entre el total del tráfico ($TP+FP+FN+TN$). Permite conocer que tan correcto es el modelo para detectar el tráfico anómalo del tráfico normal.

$$ACC = \frac{\text{predicciones correctas}}{\text{tráfico total}} = \frac{TP + TN}{TP + FP + FN + TN}$$

- Clasificación errónea (*Misclassification Rate*, MCR). Es la razón de la sumas de las predicciones incorrectas del tráfico anómalo y normal (FN + FP), entre el total del tráfico ($TP + FP + FN + TN$). Permite estimar que tan erróneo clasificó el modelo al tráfico.

$$MCR = \frac{\text{predicciones incorrectas}}{\text{tráfico total}} = \frac{FP + FN}{TP + FP + FN + TN}$$

- Coeficiente de Correlación de Matthews (*Matthews Correlation Coefficient*, MCC). Es una medida que indica la calidad de un clasificador binario, es un coeficiente que relaciona las clasificaciones reales o actuales y las predicciones del modelo. Sus valores están entre -1 y $+1$, en donde $+1$ indica una predicción perfecta, 0 para una predicción aleatoria, y -1 indica una predicción pésima, es decir, sin ninguna correspondencia entre el valor real y la predicción.

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

5.4.1. Curvas ROC

Las curvas ROC (del inglés *Receiver Operating Characteristic*) se desarrollaron en los años cincuenta como herramienta para el estudio de detección e interpretación de señales de radar [Burgueño et al., 1995], actualmente se utilizan en un gran número de áreas como en la identificación de anomalías en el tráfico de red, para diagnósticos en medicina, en inteligencia artificial, entre otras. En la detección de intrusos, la curvas ROC permiten representar de forma gráfica la relación entre la Tasa de Verdaderos Positivos (TPR) o Sensibilidad, y la Tasa de Falsos Positivos (FPR) que se calcula como $(1 - Especificidad)$. El eje x representa a la Tasa de Falsos Positivos, mientras que el eje y representa a la Tasa de Verdaderos Positivos, figura 5-3.

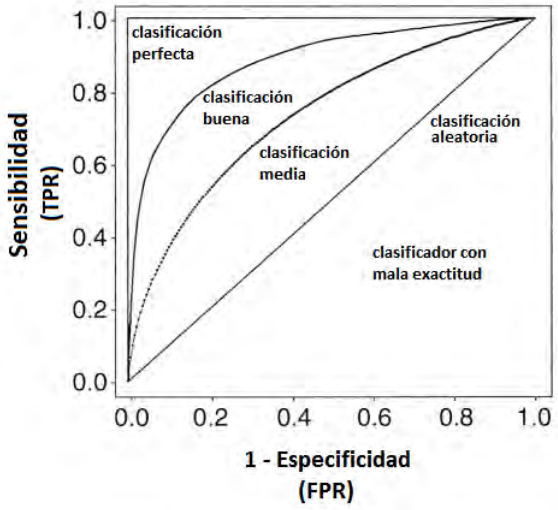


Figura 5-3: Curva ROC

Las siguientes convenciones tomadas de [Bhattacharyya and Kalita, 2014] para una curva ROC son usadas para representar la exactitud de un clasificador en el plano xy.

- El punto (0,0) indica que el modelo es exacto en clasificar el tráfico normal, sin presentar falsas alarmas [1]. Se identificará todo el tráfico como normal, sin embargo no detectará ningún comportamiento anómalo.

- El punto (1,1) indica que el modelo generará una alarma para todo el tráfico que reciba, tendrá una tasa de detección y de falsas alarmas del 100 %.

- La recta que conecta a los puntos (0,0) y (1,1) significa que el modelo clasificará con 50 % tanto para la clase del tráfico anómalo, como para la clase del tráfico normal, es por eso que se considera un clasificador aleatorio. En la práctica, la curva ROC con el comportamiento descrito, siempre se situara por debajo de esta línea.

- El punto (0,1) representa al clasificador perfecto, es decir una detección del tráfico anómalo del 100 % y una tasa de falsas alarmas de 0 %.

[1] En el concepto de Falsas Alarmas se incluyen tanto Falsos Positivos como Falsos Negativos

Por lo tanto, la exactitud de un clasificador se considerará buena, si la curva ROC comienza cerca del punto (0,0) y se dirige hacia el punto (1,1), pero en el proceso permanece cerca de los valores $x = 0$ y $y = 1$.

Los Sistemas de Detección de Intrusos o IDS actuales no son tan precisos como se desearía idealmente, muchos de ellos generan aún una gran cantidad de falsas alarmas [Bhattacharyya and Kalita, 2014], es por eso que se necesita afinar el modelo de detección para disminuir en la medida de lo posible estas falsas alarmas, estas métricas ayudan a evaluar un IDS para determinar su capacidad de detección.

En el contexto de los Sistemas de Control Industrial (ICS), si se eligen bloquear las conexiones anómalas en lugar de sólo alertarlas, es necesario que el clasificador haya sido probado y ajustado exhaustivamente, para que no se bloquee inadvertidamente una conexión normal o legítima, debido a la criticidad de estos Sistemas este bloqueo podría repercutir de manera significativa.

Capítulo 6

Obtención de datos

6.1. Resumen

Una de las principales dificultades para llevar a cabo este trabajo fue la obtención de los datos, es decir, tráfico de red de Sistemas de Control Industrial (ICS) que utilice a Modbus como protocolo de comunicación, esto porque en la mayoría de las empresas y entidades son clasificados como datos confidenciales debido a su nivel de criticidad. Se consideraron diferentes fuentes para obtenerlos, en este capítulo se revisan algunas de las fuentes cuyos datos fueron analizados y finalmente fueron descartados. Este conocimiento adquirido se usó para diseñar una infraestructura de pruebas propia con la que finalmente se obtuvieron los datos a analizar, misma que también se presenta. Con este último punto se cumple uno de los objetivos de este trabajo de investigación el cual es obtener un entorno de pruebas de Sistemas de Control Industrial que utilice a Modbus como protocolo de comunicación, de bajo costo, escalable, de hardware y software abierto.

6.2. Fuentes de datos de ICS

Se estudiaron diferentes fuentes de obtención de datos, a continuación se revisan las más relevantes.

- Se usaron los simuladores Modbus Poll y Modbus Slave desarrollados por Witte Software como parte de sus productos Modbus Tools [Witte Software, 2015]. Ambos son interfaces gráficas desarrolladas para sistemas Windows, emulan la conexión entre un maestro y varios dispositivos que actúan como esclavos. Puede emular el protocolo Modbus sobre TCP/IP. El objetivo del software es poder realizar pruebas a sistemas reales que usen el protocolo Modbus. Para las pruebas de este trabajo, se conectaron entre sí dos máquinas virtuales con sistema operativo Windows 7, en una se instaló el maestro (Modbus Poll) y en otra el esclavo (Modbus Slave), una vez establecida la configuración de red, se configuró al maestro para que enviara solicitudes (paquetes *Request*) y recibiera respuestas (paquetes *Response*) por parte del esclavo, es decir para llevar a cabo transacciones de Modbus; además se ejecutó el analizador de protocolos Wireshark para capturar el tráfico TCP/IP de la comunicación entre ambos dispositivos. Aunque se obtuvo una captura de tráfico de red con el protocolo Modbus TCP/IP simulando una conexión entre dos dispositivos de un ICS, se determinó que no sería útil al propósito de la investigación debido a que los datos que el esclavo envía

al maestro debían ser introducidos manualmente por el usuario, entonces se debían tener datos previos de algún ICS para poder emular un comportamiento apegado a la realidad. Además, el software está limitado en la generación de datos de Modbus, dado un valor inicial, por ejemplo para algún registro discreto de entrada o salida, solamente se permite mantener o incrementar el valor en una unidad, no así de decrementarlo, ni tampoco permite modificar los valores de incremento o de decremento a más de una unidad. Únicamente permite conexiones entre el maestro y los esclavos de 10 minutos de duración, después de este tiempo es necesario reiniciar los programas. Se descartó esta fuente para la obtención de datos.

- Se probó el generador de tráfico Modbus usado para evaluar la seguridad en sistemas SCADA utilizado en [Al-Dalky et al., 2014]. Esta herramienta programada en Python con bibliotecas del generador de tráfico Scapy, genera tráfico Modbus TCP/IP a partir de la entrada de reglas del IDS Snort. Las reglas de Snort son patrones que han sido creadas para la detección de tráfico malicioso, como se explicó en el Capítulo 5 para crearla es necesario que previamente se identifique el patrón de tráfico malicioso por lo general son creadas por expertos en el área de detección de intrusos en el tráfico de red. Entonces, la herramienta genera tráfico malicioso el cual es enviado a los dispositivos que forman parte de un ICS para conocer su comportamiento ante estos ataques. Aunque se puede obtener una captura de tráfico de red con el protocolo Modbus TCP/IP, es necesario contar previamente con las reglas de Snort, por lo que generar tráfico de esta forma se asemeja más al modelo de detección de intrusos basados en firmas o reglas. Este método está limitado frente a la aparición de nuevos vectores de ataque que no han sido detectados, por lo tanto no han sido modelados y no se ha creado una regla que pueda detectarlos, por este motivo se descartó obtener tráfico de red de ICS con este método.

- Se encontró una captura de tráfico de una red ICS en el repositorio CloudShark [Cloudshark, 2012], ésta fue subida por una persona que en el foro de preguntas y respuestas del analizador de protocolos Wireshark describía un problema que se presentaba en la captura, el cual era una desconexión atribuida probablemente al tiempo en que el dispositivo Maestro tardaba en procesar las respuestas provenientes de un dispositivo esclavo [Ask Wireshark, 2012]. A pesar de que el tráfico de red correspondía a un sistema ICS real con datos de una operación normal, se descartó su uso debido a que la única anomalía que se presentaba era la desconexión, es decir, no se contaban con suficientes datos anómalos para poder realizar el análisis de detección de anomalías. Aún así se realizaron los primeros análisis con este tráfico tomando secuencias de tráfico de trece paquetes de respuesta o *Response* de Modbus, se tomaba el paquete de respuesta siempre que hubiera habido un paquete de solicitud o *Request*, eran 13 dispositivos esclavos de la captura tráfico, y las dos características que se usaron como variables fueron la suma del tamaño del payload de cada paquete *Response* en una secuencia medido en bytes, y el tiempo de respuesta del paquete *Response* ante un paquete *Request*, esto en un intento de modelar todo el comportamiento de la red, como datos maliciosos se añadieron secuencias anómalas generadas aleatoriamente. Se muestra en la figura 6-1 un diagrama de la conexión entre los dispositivos esclavos (PLC) y el maestro.

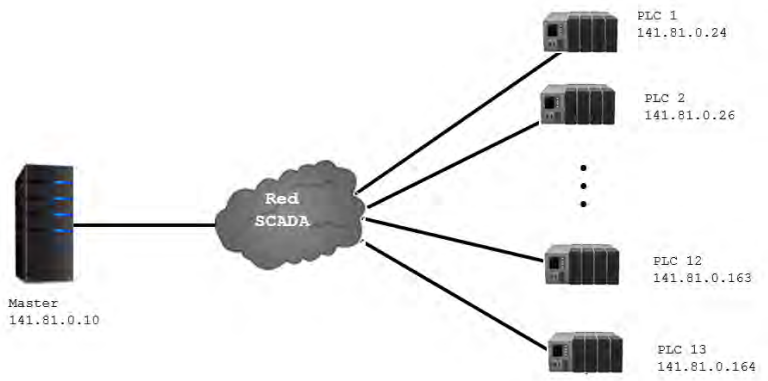


Figura 6-1: Diagrama lógico de la infraestructura de la captura de tráfico de CloudShark.

- Basado en la propuesta de [Bhatia et al., 2014] se hizo la prueba de obtener tráfico de red usando el equipo PLC Compact RIO modelo 9074 de National Instruments, del Laboratorio de Control de la Fac. de Ingeniería. El Compact RIO actuó como dispositivo para adquirir datos de un circuito eléctrico, obtener datos tanto de entradas analógicas como digitales. En el maestro se tenía el software Lab View configurado con el modulo para el protocolo Modbus, tanto el maestro como el esclavo se conectaron a un switch de red a través de la pila de protocolos TCP/IP. El diagrama de conexión se muestra en la figura 6-2 junto con imágenes de los equipos utilizados en la figura 6-3. Se descartó esta forma de obtención de datos debido a que el tiempo disponible para trabajar con el Compact RIO era limitado, además de que era recomendable contar con al menos otro dispositivo de adquisición de datos para obtener mayor tráfico de red y obtener mejores resultados en el análisis de éstos.

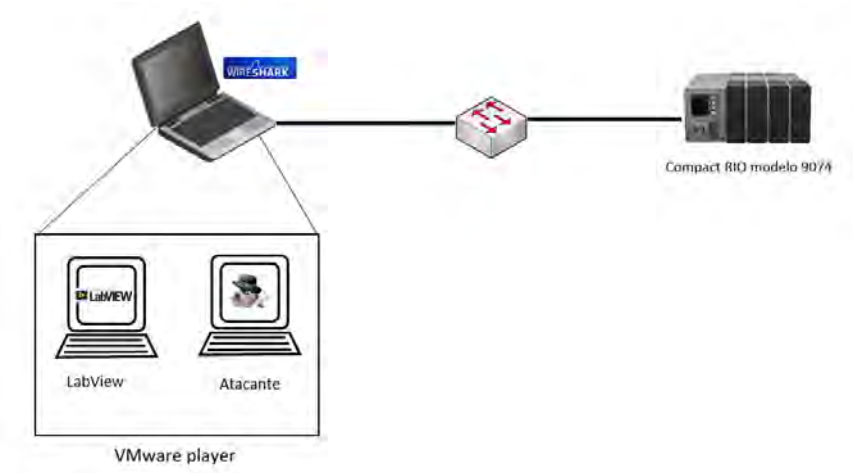
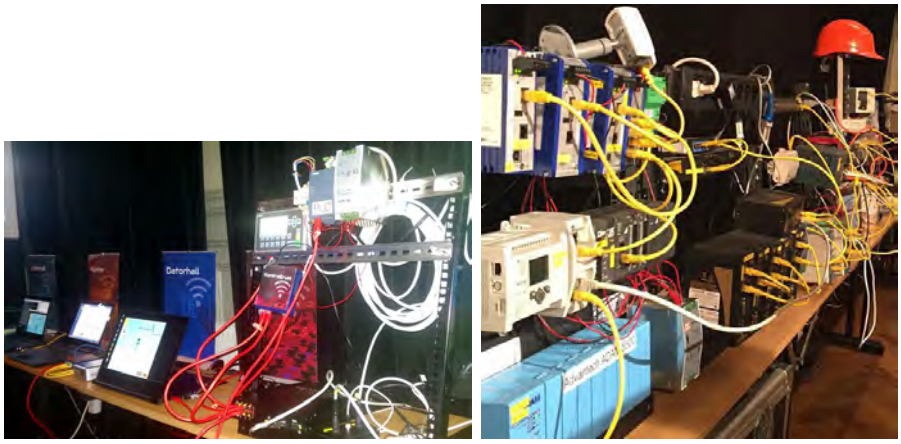


Figura 6-2: Diagrama lógico de la infraestructura de la captura de tráfico usando el Compact RIO 9074.

(a) Compact RIO 9074 (b) Infraestructura de pruebas

Figura 6-3: Infraestructura para el Compact RIO 9074.

- En octubre de 2015 se llevó a cabo en Estocolmo, Suecia una junta sobre ciberseguridad en ICS y SCADA llamada 4SICS [Hjelmvik, 2015]. Durante este evento se colocó un laboratorio con dispositivos de ICS como PLC, RTU, servidores, y equipo de interconexión de redes (switch, router, firewall, etc); durante los tres días de duración del evento, se pusieron a disposición de los asistentes para que fueran atacados y de esa forma probar la seguridad. Estas capturas de tráfico fueron puestas a disposición de cualquier persona. Se descartó también el uso de estos datos debido a que si bien si existía tráfico anómalo, los dispositivos esclavos no monitoreaban algún Sistema de Control Industrial, entonces no había tráfico de red que fueran capturado con comportamiento normal, esta conclusión fue confirmada por Erik Hjelmvik quien fue el encargado de configurar la infraestructura. En la figura 6-4 se muestran fotografías del laboratorio ICS en 4SICS.



(a) Red demo de Svenska Kraftnät (b) Laboratorio de ICS

Figura 6-4: Infraestrucura ICS para 4SICS.

Si bien los datos de las fuentes revisadas no se utilizaron para este trabajo, esto permitió contar con un conocimiento que derivó en el diseño y creación de la infraestructura con la que se obtuvo el tráfico de red a analizar.

6.3. Infraestructura de ICS para pruebas

Se desarrolló una infraestructura sencilla de ICS usando componentes de fácil programación y adquisición, con la que se capturó tanto tráfico normal del monitoreo de circuitos electrónicos, como tráfico anómalo generado a partir de un programa de suplantación de datos de ICS, ambos sobre el protocolo Modbus TCP/IP. Se basó en el tutorial técnico presentado en FleaPLC [2014]. En la figura 6-5 se muestra dicha infraestructura.

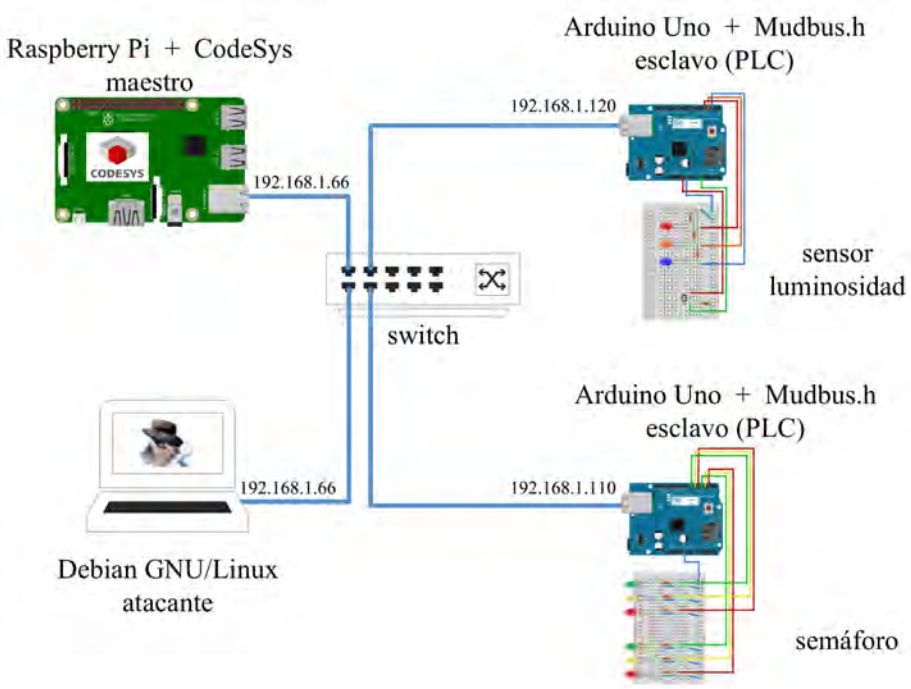


Figura 6-5: Diagrama lógico de la infraestructura de ICS de pruebas.

A continuación se describen los componentes de la infraestructura de pruebas,

- Raspberry Pi. Es una computadora de placa reducida (o *single-board computer*). Los modelos que existen son Pi-1 modelos A+ y B+, Pi-2 modelos B, Pi-3 modelo B y Pi Zero. Para este trabajo se utilizó la Raspberry Pi-1 modelo B cuyas características son procesador ARM 1176 a 1 GHz, 512 MB en RAM, dos puertos USB, salida HDMI y Ethernet. Página web `https://www.raspberrypi.org`.

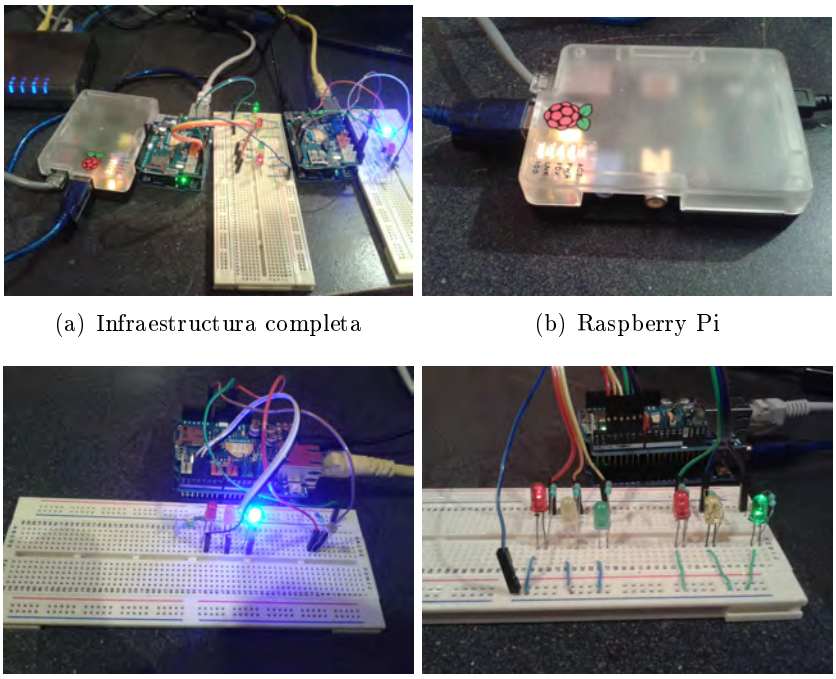
- CODESYS. Software para ICS, el cual es ampliamente usado para la creación y ejecución de programación a través de diagramas lógicos para automatización de procesos industriales, en el año 2012 este software era utilizado por alrededor de 261 fabricantes [Digital Bond, 2012]. Página web `https://www.codesys.com`.

- Arduino Uno. Plataforma de prototipos de electrónica de código abierto, basada en hardware y software fáciles de usar. Está confonformada por un microcontrolador ATmega328P a 16 MHz y 32 KB de memoria. No ejecuta un sistema operativo propio, si no que el microcontrolador se programa directamente. Además se utilizó el módulo para conexiones Ethernet. Página web `http://www.arduino.org`.

- Biblioteca Mudbus. Biblioteca para configurar un dispositivo esclavo en Arduino operando bajo el protocolo Modbus TCP/IP, está limitada en cuanto al número de funciones de Modbus. Se realizó una mínima modificación para que pudiera operar con la versión 2 del módulo de Ethernet para Arduino. Página web `https://github.com/luizcantoni/mudbus`.

- Circuitos - Semáforo y Sensor de luminosidad. Se diseñaron dos circuitos electrónicos para enviar y recibir datos del maestro (Raspberry Pi). El primer circuito corresponde a un sensor que mide la luminosidad del ambiente y con base en estos datos recibe una señal para encender cualquiera de los tres led. El segundo circuito es un semáforo para un cruce de calles, es decir cuenta con dos semáforos sincronizados, solamente recibe la señal para encender los led. En el Apéndice C se muestran los diagramas de los circuitos eléctricos.

Se muestran fotografías de la infraestructura de pruebas en la figura 6-6.



(a) Infraestructura completa (b) Raspberry Pi

(c) Sensor de luminosidad (d) Semáforo

Figura 6-6: Infraestructura de pruebas.

El dispositivo maestro tiene instalado un programa de control remoto especialmente diseñado para Raspberry Pi que forma parte de la suite de CODESYS, con esta instalación se pudo cargar el programa para controlar, enviar y recibir datos a través del protocolo Modbus TCP/IP hacia los dispositivos esclavo, es decir a los Arduino Uno, este programa se realizó con el software CODESYS versión 3.5. Estos dos dispositivos esclavos corresponden a los PLC, fueron programados con el IDE de Arduino para que enviaran señales de control a los dos circuitos, al sensor de luminosidad y al semáforo. Además en el programa de Arduino se incluyó a la biblioteca de funciones Mudbus.h para poder enviar y recibir datos hacia el programa de control remoto de CODESYS alojado en la Raspberry Pi, usando como protocolo de comunicación a Modbus TCP/IP. Los Arduino Uno envían a los circuitos señales de

control (voltajes) para encender o apagar los led, y reciben la información de la intensidad de luz del circuito del sensor de luminosidad, esto lo hacen a través de sus entradas y salidas digitales y analógicas; por otro lado, envían al dispositivo maestro los datos de la intensidad luminosa y reciben de éste los datos para encender o apagar los led, esto lo realizan enviando y recibiendo paquetes del protocolo Modbus TCP/IP a través de la interfaz de Ethernet. Es por eso que a los Arduino Uno se les configuró en el mismo programa de control, una dirección MAC, y una dirección IP del mismo segmento de red que el maestro. Para realizar la conexión de red entre los dos dispositivos esclavos y el maestro se hace uso de un switch. En el Apéndice B se detalla el funcionamiento de los Arduino Uno como PLC, cada uno de los circuitos electrónicos y cómo se envía y recibe la información de control usando el protocolo Modbus TCP/IP.

El equipo atacante corresponde a un equipo virtualizado con sistema operativo Debian GNU/Linux conectado a la misma red de la infraestructura de pruebas, desde el cual se ejecutó un programa escrito en el lenguaje C cuya función es enviar paquetes maliciosos hacia los dispositivos esclavos haciendo un ataque de suplantación del dispositivo maestro. El programa se ejecuta con el comando,

```
./gen_mb_traffic saddr daddr sport func_code unit_id mb_data_file
```

en donde

- `gen_mb_traffic` - nombre del programa
- `saddr` - dirección IP a suplantar, la misma dirección IP del maestro
- `daddr` - dirección IP a la que se enviarán los paquetes maliciosos, en un ejecución del programa será para el PLC conectado al circuito semáforo y en otra ejecución será para el PLC conectado al sensor de luminosidad
- `sport` - puerto TCP origen del dispositivo que inicia la conexión, el mismo puerto origen de la conexión del maestro hacia el esclavo
- `func_code` - mismo código de función usado en los paquetes Modbus de una conexión entre maestro y esclavo
- `unit_id` - mismo número de identificación usado en los paquetes Modbus de una conexión entre maestro y esclavo
- `mb_data_file` - archivo binario con los datos a enviar hacia los esclavos

Se programaron las conexiones TCP usando las bibliotecas del lenguaje C para la creación de sockets, se programaron también los paquetes de la capa de aplicación siguiendo la especificación del protocolo Modbus tratada en el Capítulo 3. Es importante señalar que se analizaron dos formas para suplantar la dirección IP del maestro, la primera consiste en formar los paquetes de los protocolos IP y TCP en lugar de dejar esta tarea al sistema operativo, debido al interés de mantener la conexión entre el equipo atacante y los esclavos también es necesario programar el comportamiento de los protocolos IP y TCP para poder mantener esta comunicación, por ejemplo se tiene que programar el control de flujo y congestión para el protocolo TCP, debido al tiempo que requería esta tarea se descartó esta opción; la segunda forma consiste en asignar la dirección IP del maestro como una dirección IP alias a la interfaz de red del atacante, de esta forma la programación de la herramienta de ataques se facilitaba al solamente tener que programar desde cero la capa de aplicación.

Por otro lado, el haber programado la herramienta de ataques solamente para enviar paquetes implicaría el establecer nuevas conexiones con el esclavo por cada paquete Modbus a enviar, esto provocaría que el atacante fuera fácilmente detectado por cada negociación de la conexión a través del Three Way Handshake, aún suplantando el número de puerto origen del maestro. También se configuró el sistema operativo del equipo atacante para que tuviera el mismo número inicial del tamaño de ventana del protocolo TCP que el maestro.

Una vez teniendo la infraestructura de pruebas y al equipo atacante, se procedió a obtener la captura de tráfico de red. Involucra tres tipos de ataques descritos en el Capítulo 4, primero un ataque de Reconocimiento Pasivo (o *Passive reconnaissance*) usado para obtener datos de la red tanto del maestro como de los esclavos, como por ejemplo direcciones IP, números de puerto TCP, tamaño inicial de ventana de TCP, cabecera y datos del protocolo Modbus, velocidad de transmisión de los paquetes, entre otros, con el fin de ser suplantados por el atacante. También se usaron conceptos de los ataques *Rogue interloper*, que si bien el atacante no está en una red externa, sí fabrica mensajes de Modbus; y *Direct slave control*, que aunque no se bloquean los mensajes del maestro hacia el esclavo, cada vez que se envía un paquete Modbus malicioso se sobrescribe la acción del maestro sobre el esclavo con la acción del atacante sobre el esclavo, esto permanece hasta que el maestro envía de nuevo un paquete normal o legítimo al esclavo.

Usando el programa gen_mb_traffic descrito con anterioridad junto con los cambios a las configuraciones en el sistema operativo del atacante, se usaron los siguientes parámetros,

- Dirección IP del maestro,
- Número de puerto de la conexión TCP del maestro hacia los dos esclavos, un número de puerto diferente para cada conexión con los PLC esclavos,
- Número inicial de tamaño de ventana TCP del maestro,
- Identificador de Unidad de cada dispositivo esclavo, diferente para cada esclavo, en la cabecera del protocolo Modbus,
- Código de Función de los PLC esclavos, en la Unidad de Datos (PDU) del protocolo Modbus,
- Datos asociados al tipo de Función y comportamiento de cada PLC esclavo, en la Unidad de Datos (PDU) del protocolo Modbus,
- Se trato de igualar la velocidad de transmisión de los paquetes del atacante hacia los esclavos con la del maestro hacia los esclavos.

Tanto en el maestro como en el esclavo se ejecutó el analizador de protocolos Tcpdump para capturar el tráfico de red del protocolo Modbus. De esta forma se obtuvieron dos capturas de tráfico de red, una con la comunicación entre el dispositivo maestro y los esclavos, y la otra captura con la comunicación entre el atacante y los esclavos.

Capítulo 7

Análisis de los datos

7.1. Resumen

Una vez que se tienen las capturas de tráfico con conexiones anómalas y normales es necesario procesarlo para obtener la información del tráfico en un formato que pueda ser utilizado para aplicar el clasificador ingenuo de Bayes. Se decidió utilizar el tráfico en flujos para este análisis, se obtendrán las características de cada flujo como direcciones IP, duración del flujo, número de paquetes transmitidos, cantidad de bytes transmitidos, entre otras, además se convierten los valores cotinuos de cada característica en valores discretos usando la técnica de *Coarse grain*. Posteriormente se detalla cómo se aplicó el clasificador ingenuo de Bayes a los flujos obtenidos, se muestran las variables más relevantes respecto a su valor de *Score* y se discuten los resultados. Así pues, con lo presentado en este capítulo se llevan a cabo los dos últimos objetivos propuestos al inicio de este trabajo, el primero es, aplicando el clasificador ingenuo de Bayes, obtener las variables de mayor influencia a través de su valor de *Score*, que contribuyan a determinar si un flujo del tráfico de red previamente capturado es anómalo o no, el segundo objetivo es, con base en métricas de evaluación, determinar si es viable utilizar el clasificador ingenuo de Bayes como método de detección de anomalías en un NIDS para el entorno de pruebas.

7.2. Procesamiento de los datos

Como se explicó en el Capítulo 6, se generaron dos capturas de tráfico de red, una proveniente del equipo atacante y la otra con la comunicación normal o legítima entre los esclavos (PLC construidos con Arduino) y el maestro (computadora Raspberry con el software CODESYS). Se usó el software de generación de flujos de red Argus para obtener los datos a analizar. Argus es una herramienta que a partir de paquetes de red genera flujos de red con base en las transacciones de los paquetes, se usa tanto en el procesamiento de tráfico de red en tiempo real como en archivos de tráfico de red previamente generados [Bullard, 2007].

En la captura de tráfico de red el atacante llevó a cabo la suplantación de algunos parámetros del maestro, los cuales fueron descritos en el Capítulo 6. En las tablas 7.1 y 7.2 se muestran los datos generales de las capturas de tráfico, en cada captura sólo están presentes los protocolos IP, TCP y Modbus, de las capas de red, transporte y aplicación del Modelo TCP/IP, respectivamente. La captura de tráfico normal se inició antes y se terminó después de la captura de tráfico anómalo con el fin de tener una línea base del

comportamiento normal del sistema, por otra parte, los ataques no se llevaron a cabo de forma continua, en el periodo de la captura de tráfico anómalo se distinguen dos intervalos no continuos de ataques.

Captura de tráfico con anomalías	
Nombre del archivo	gen_traffic.pcap
Número de paquetes	46000
Duración de la captura	6448 segundos (107.46 min)
Fecha de inicio	30-12-2016 16:27:32
Fecha de término	30-12-2016 18:15:00
Velocidad	3600 bits/s (450 bytes/s)
Tamaño promedio de paquete	62 bytes
Velocidad promedio de paquetes	7 paquetes/s

Tabla 7.1: Características de la captura de tráfico anómalo

Captura de tráfico normal	
Nombre del archivo	rasp_traff.pcap
Número de paquetes	91000
Duración de la captura	8189 segundos (136.48 min)
Fecha de inicio	30-12-2016 16:18:29
Fecha de término	30-12-2016 18:34:58
Velocidad	5654 bits/s (706 bytes/s)
Tamaño promedio de paquete	63.12 bytes
Velocidad promedio de paquetes	11 paquetes/s

Tabla 7.2: Características de la captura de tráfico normal

En la figura 7-1 se muestran el flujo de bits por segundo de la captura de tráfico anómalo y normal, los incrementos de tráfico que se observan corresponden a los dos intervalos en los que se ejecutó el ataque.

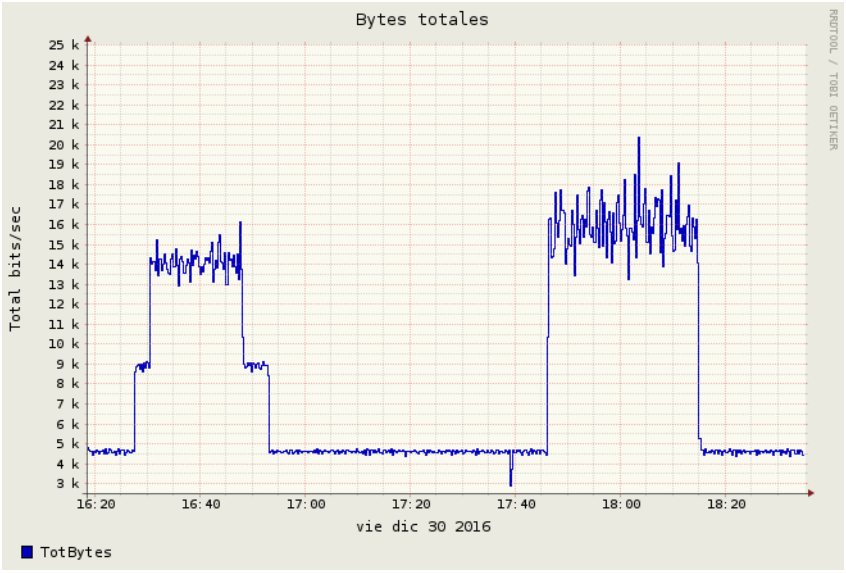


Figura 7-1: Bytes totales

Con la herramienta de flujos de red Argus se generaron por cada captura de tráfico un archivo de flujos de red, se usaron las siguiente opciones del comando `argus`:

- `A`, genera métricas en bytes de la capa de aplicación por cada registro generado de la captura de tráfico,
- `J`, genera datos del rendimiento de los paquetes en el tráfico de red, por cada registro generado a partir de la captura de tráfico,

Posteriormente con el cliente Ra de Argus se extrajeron los atributos o variables de los flujos de red en formato de texto y se exportaron a un archivo separado por comas (Comma Separated Values). Se realizó una discriminación de variables de los flujos, se descartaron las variables que tuvieran dependencia entre sí, aquellas que fueran generadas por alguno de los protocolos de red y que no fueran predecibles fácilmente para poder obtener una probabilidad de su presencia en el flujo, como por ejemplo los números de secuencia del protocolo TCP. En el Apéndice D en la Tabla D.3 se explica porqué algunas de las variables generadas a partir del archivo de flujos de red, fueron descartadas. De forma similar, en el mismo Apéndice D en las Tablas D.1 y D.2 se detalla cada una de las variables empleadas para el análisis.

Después de la generación de los archivos de los datos que se usarán en el clasificador ingenuo de Bayes, se agregó una variable más que corresponde al etiquetado de los datos, si los datos del flujo provienen del tráfico anómalo se etiquetan como Positivos, en cambio si los datos del flujo provienen del tráfico normal se etiquetan como Negativo. En un sólo archivo se juntaron los datos de la captura de tráfico de red anómalo y normal ordenados respecto al tiempo.

Es necesario antes de comenzar a usar el modelo ingenuo de Bayes hacer una conversión de las variables de los flujos que tengan valores continuos, a valores discretos. Esto se realiza estableciendo conjuntos en los cuales se define un rango de valores para ese conjunto, a los valores continuos se les asignará la pertenencia a un conjunto dependiendo del rango en el que se encuentre. Para establecer los rangos de pertenencia, se toman en cuenta el valor mínimo y máximo de la variable, su promedio, su desviación estándar y en gran medida la descripción de la variable. De la misma forma se asigna un valor numérico entero para las variables cuyos valores son de tipo texto. En el Apéndice D se muestra la tabla de valores discretos para las variables continuas. Esta técnica se conoce con el nombre de *Coarse grain*.

Por último se divide el archivo de los datos de los flujos de forma aleatorio en una proporción aproximada de 70 % que serán los datos de entrenamiento del clasificador y de 30 % que serán los datos de prueba, tabla 7.3. Cada flujo a analizar se conforma de 39 variables.

Archivo	**Descripción**	**Flujos**	**Anómalos**	**Normales**
Train.csv	Datos de entrenamiento	3061	823	2238
Test.csv	Datos de prueba	1187	354	833

Tabla 7.3: Archivos de entrenamiento y prueba

7.3. Análisis

Una vez que se tienen los datos de los flujos o registros etiquetados respecto a su pertenencia a la clase de tráfico con anomalías o a la no clase, es decir, al tráfico normal, y que los valores continuos han sido transformados en discretos, se realiza el cálculo de las probabilidades de las variables para determinar cual de ellas es relevante en términos de la predictibilidad de pertenencia o no a la clase de tráfico con anomalías. Se realiza este cálculo con base en lo descrito en el Capítulo 5. Al aplicar los cálculos se obtienen por cada variable y valor $var_j[x_i]$ entre otros resultados, su puntuación o *Score* con el fin de conocer que tanto influye dicha variable y valor en la predicción de pertenencia a la clase del tráfico anómalo. Para las siguientes explicaciones se usará la notación $var_j[x_i]$ para indicar que la variable var_j tiene el valor x_i, por ejemplo $SrcBytes[1]$ indica que se trata de la variable *SrcBytes* con valor 1.

En la tabla 7.4 se muestra a modo de ejemplo las primeras cinco variables y su valor, con mayor *Score*. Para la variable $SrcPkts[3]$ se tiene un valor de *Score* de 6.01, el más alto para los datos de entrenamiento, esto se debe a que de los 301 registros que presentan esta variable, todos pertenecen a la clase de tráfico anómalo y ninguno a la clase de tráfico normal, por lo tanto se incrementa su puntuación o *Score*; el valor de 3 indica que se enviaron de 15 a 20 paquetes de acuerdo a la tabla D.5, y de acuerdo a la tabla D.1 por el nombre de la variable se conoce que se enviaron los paquetes desde el origen hacia el destino, este rango de paquetes enviado no se presentó en ninguno de los 2238 registros de tráfico normal, solamente en 301 de los 823 registros del tráfico anómalo. En cuanto a las variables $SrcBytes[9]$, $SrcBytes[11]$ y $SrcBytes[8]$, indican que se enviaron desde el origen de 900 a 1000 bytes, de 1100 a 1200 bytes y de 800 a 900 bytes respectivamente, estos rangos de bytes enviados tampoco se presentaron en alguno de los registros de tráfico normal, es por eso que también tienen un *Score* alto. De forma similar, respecto a la variable $SIntPkt[3]$, de los 272 registros con presencia de esta variable 270 pertenecen a la clase de tráfico anómalo, mientras que sólo 2 pertenecen a la clase de tráfico normal, de aquí su Score elevado, el significado de la variable corresponde al tiempo de llegada entre paquetes del origen en un rango de 300 a 400 milisegundos.

Variable	*Valor*	Nx	Ncx	$N\bar{c}x$	Nc	$N\bar{c}$	N	$P(c)$	$P(\bar{c})$	$P(c \mid x)$	$P(x \mid c)$	*Epsilon*	*Score*
SrcPkts	3	301	301	0	823	2238	3061	0.27	0.73	1	0.37	28.61	6.01
SIntPkt	3	272	270	2	823	2238	3061	0.27	0.73	0.99	0.33	26.92	5.21
SrcBytes	9	128	128	0	823	2238	3061	0.27	0.73	1	0.16	18.66	5.16
SrcBytes	11	109	109	0	823	2238	3061	0.27	0.73	1	0.13	17.22	5
SrcBytes	8	94	94	0	823	2238	3061	0.27	0.73	1	0.11	15.99	4.85

Tabla 7.4: Ejemplos de variables con *Score* positivo

En la tabla 7.5 se muestran algunas variables y valores con *Score* cero y cercanos al cero, lo cual indica que no son muy útiles para la predicción. Las variables $dIpId[3]$, $sIpId[0]$, $dIpId[12]$, $dIpId[2]$ y $sIpId[10]$ con *Score* de 0.01, 0.01, 0, -0.01 y -0.02 respectivamente, están asociadas al número de identificación de la cabecera del protocolo IP, este campo identifica de forma única a los paquetes IP y es generado de forma pseudoaleatoria por el sistema operativo en un rango de 0 a 65535 [Northcutt and Novak, 2002], es por eso que es difícil relacionar los números de identificación con el tráfico anómalo o normal, a menos que este número sea repetitivo en los paquetes. Las variables $Proto[1]$ y $Dport[502]$ ambas con valor de *Score* de 0, significan que en el flujo se usa el protocolo TCP y Modbus, tienen

ese valor de *Score* debido a que son los únicos protocolos que se utilizan tanto en el tráfico anómalo como en el tráfico normal, es por eso que no contribuyen a la predicción. La variable *SrcAddr*[3232235842] corresponde a la dirección IP del maestro que ha sido suplantada por el atacante, su *Score* de 0 no contribuye a la predicción debido a que todos los flujos de tráfico anómalo y de tráfico normal tienen como origen esta dirección IP.

Variable	*Valor*	Nx	Ncx	$N\bar{c}x$	Nc	$N\bar{c}$	N	$P(c)$	$P(\bar{c})$	$P(c \mid x)$	$P(x \mid c)$	*Epsilon*	*Score*
dIpId	3	219	60	159	823	2238	3061	0.27	0.73	0.27	0.07	0.17	0.01
sIpId	0	242	66	176	823	2238	3061	0.27	0.73	0.27	0.08	0.14	0.01
SrcAddr	3232235842	3061	823	2238	823	2238	3061	0.27	0.73	0.27	1	0	0
Proto	1	3061	823	2238	823	2238	3061	0.27	0.73	0.27	1	0	0
Dport	502	3061	823	2238	823	2238	3061	0.27	0.73	0.27	1	0	0
dIpId	12	237	64	173	823	2238	3061	0.27	0.73	0.27	0.08	0.04	0
dIpId	2	226	61	165	823	2238	3061	0.27	0.73	0.27	0.07	0.04	-0.01
Flgs	1	3055	817	2238	823	2238	3061	0.27	0.73	0.27	0.99	-0.18	-0.01
sIpId	10	240	61	176	823	2238	3061	0.27	0.73	0.27	0.08	-0.08	-0.02

Tabla 7.5: Ejemplos de variables con *Score* cercano al cero

Por último, en la tabla 7.6 se muestran algunas variables y valores con *Score* negativo. La variable *DstLoad*[10] presenta el *Score* más bajo de las variables de los datos de entrenamiento con un valor de -8.14, esto se debe a que de los 450 registros que presentan esta variable todos corresponden al tráfico normal y ninguno a tráfico anómalo, es por eso que este valor tiene una predicción alta para indicar que un registro pertenece a la no clase de tráfico anómalo, esta variable se refiere a los bits por segundo enviados por el destino en un rango 1000 a 1100. De forma similar las otras variables presentadas en la tabla presentan un *Score* negativo que permite predecir la pertenencia del registro a la no clase de tráfico anómalo, es decir que pertenecen al tráfico normal.

Variable	*Valor*	Nx	Ncx	$N\bar{c}x$	Nc	$N\bar{c}$	N	$P(c)$	$P(\bar{c})$	$P(c \mid x)$	$P(x \mid c)$	*Epsilon*	*Score*
DstLoad	5	258	0	258	823	2238	3061	0.27	0.73	0	0	-9.74	-7.58
DstBytes	7	593	0	593	823	2238	3061	0.27	0.73	0	0	-14.77	-7.58
DstLoad	4	448	0	448	823	2238	3061	0.27	0.73	0	0	-12.84	-8.13
DstBytes	3	708	0	708	823	2238	3061	0.27	0.73	0	0	-16.14	-8.13
DstLoad	10	450	0	450	823	2238	3061	0.27	0.73	0	0	-12.86	-8.14

Tabla 7.6: Ejemplos de variables con *Score* negativo

En cada registro de los datos de prueba, que al igual que los datos de entrenamiento corresponden a un flujo con sus 39 variables, se sustituye el valor de *Score* asociado a cada variable $var_j[x_i]$, previamente calculado con los datos de entrenamiento. Una vez que los valores de *Score* para cada $var_j[x_i]$ de cada registro hayan sido sustituidos, se sumarán entre sí los *Score* de cada variable del registro dando como resultado un *Score* total por registro, este valor final es el que determinará si un registro o flujo pertenece o no a la clase de tráfico anómalo.

Después de haber llevado a cabo la clasificación de los datos de prueba se comparó la predicción llevada a cabo por el clasificador ingenuo de Bayes con el valor real. Para determinar con base en el valor de *Score* total, cuándo un registro pertenece o no a la clase, se usó el criterio ROC01, que consiste en el cálculo de la distancia euclidiana de cada punto de

la curva ROC al punto (0,1) que representa el punto de un clasificador perfecto [Valencia, 2017], el punto de corte óptimo para este análisis es el punto (0.035, 0.96), el cual representa el punto sobre la curva ROC más cercano al punto (0,1), y el valor de *Score* para dicho punto de corte es de -9.07. Si la suma total de *Score* de un registro es mayor a este valor, será considerado como anómalo, si es menor será considerado como normal.

Siguiendo las métricas de evaluación propuestas en el Capítulo 5 se obtuvieron los siguientes resultados, en la tabla 7.7 se muestra la Matriz de confusión en la que se observa que todos los flujos de tráfico anómalo fueron clasificados correctamente, es decir se obtuvieron 354 Verdaderos Positivos (TP) y no se clasificó ninguno erróneamente, cero Falsos Negativos (FN), sin embargo 228 de los 833 flujos de tráfico normal fueron clasificados como anómalos, es decir se presentaron 228 Falsos Positivos (FP), y se clasificaron correctamente 605 flujos de tráfico normal (TN).

	Verdadero (T)	**Falso (F)**
Positivo (P)	TP = 354	FP = 228
Negativo (N)	TN = 605	FN = 0

Tabla 7.7: Matriz de confusión para los flujos de pruebas

En los valores presentados en la tabla 7.8 se observa que el valor de Sensibilidad es igual a 100 %, mientras que el valor de especificidad es igual a 73 %, lo que indica que se prefirió la detección de todas las anomalías sobre la eficiencia del sistema. Se obtuvieron 39 % de falsos descubrimientos contra 61 % de precisión del clasificador. El valor de exactitud es alto al tener 81 % se clasificaciones correctas, sin embargo los Falsos positivos representan una clasificación errónea del 19 %. La calidad de este clasificador binario determinada por el Coeficiente de Matthews es de 0.66, por lo que puede considerarse como bueno, ya que la clasificación perfecta tienen una puntuación de 1, mientras que una predicción pésima tiene una puntuación de -1.

Métrica		**Valor**
Sensibilidad	(TPR)	1
Especificidad	(TNR)	0.73
Precisión	(PPV)	0.61
Falsos descubrimientos	(FDR)	0.39
Exactitud	(ACC)	0.81
Clasificación errónea	(MCR)	0.19
Coeficiente de correlación de Matthews	(MCC)	0.66

Tabla 7.8: Resultados de las métricas de evaluación

La curva ROC de la figura 7-2 sigue las características deseadas en un clasificador binario, es decir, la exactitud de un clasificador se considera buena, si la curva ROC comienza cerca del punto (0,0) y se dirige hacia el punto (1,1), pero en el proceso permanece cerca de los valores $x = 0$ y $y = 1$.

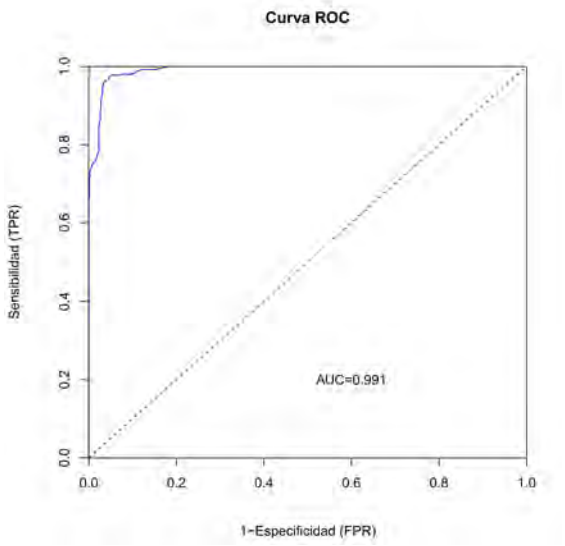


Figura 7-2: Curva ROC para el clasificador

Además se obtuvo la gráfica de la figura 7-3 de la Estimación kernel de la función de densidad entre la pertenencia a la clase real y la predicción del clasificador, para los datos de prueba. En color verde se representan a la densidad de los flujos de tráfico normal y en color azul a la densidad de los flujos de tráfico anómalo, se observa con claridad que existe un rango en que los valores de ambos se cruzan, esta área corresponde a los valores de las variables que son comunes para las dos clases, es decir a los Falsos Negativos y Falsos Positivos resultantes de una clasificación, para este análisis solamente se presentaron Falsos Positivos.

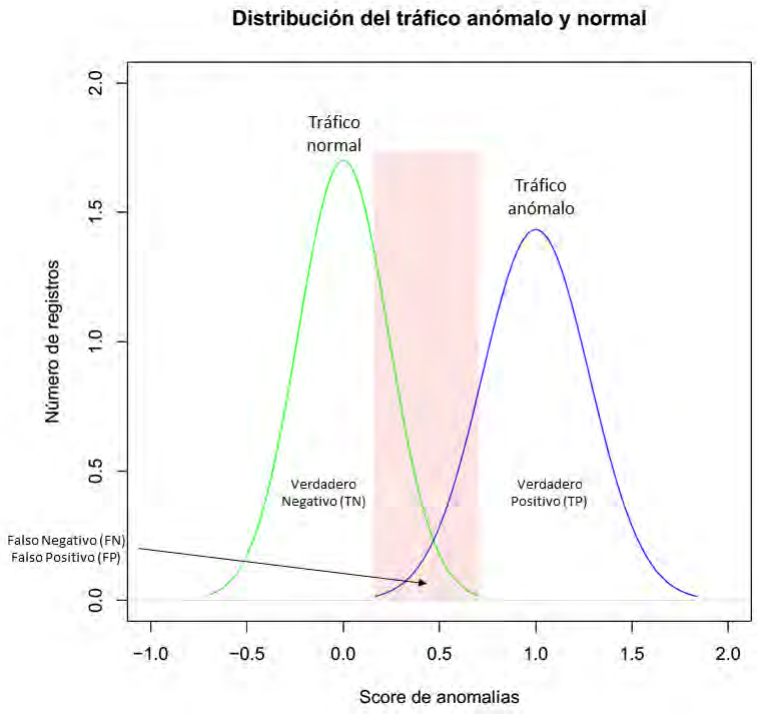


Figura 7-3: Distribución del tráfico anómalo y normal

Capítulo 8

Conclusiones

Una de las principales dificultades para la realización de este trabajo fue la obtención de tráfico de red de Sistemas de Control Industrial (ICS) con anomalías y con el comportamiento normal de una red de este tipo, es comprensible que las empresas resguarden estos datos debido al nivel de criticidad de estos. Para la tarea de clasificación se debe identificar con claridad el tráfico anómalo del normal por lo que aunque se hubiera tenido acceso a tráfico de red de ICS, también hubiera sido necesario el contar con autorización para realizar las pruebas de ataques o que quien proporcionara este tráfico tuviera al menos el conocimiento de que efectivamente en los datos también se presentaban anomalías. Lo deseable sería contar con una infraestructura de ICS diseñada especialmente para pruebas con el fin de poder realizar los ataques a la infraestructura sin temor a provocar un daño en los sistemas automatizados.

Como ventaja de la infraestructura de ICS presentada en este trabajo es el bajo costo y la configuración sencilla, se puede escalar a un mayor tamaño y se pueden agregar otros dispositivos para monitorear su comportamiento tales como motores, medidores en tanques de agua, sensores de temperatura, entre otras. Debido a que la biblioteca de funciones de Modbus para Arduino es de código abierto es posible programar más funciones del protocolo Modbus para que responda de una forma más apegada al comportamiento de los dispositivos de ICS comerciales. Se puede concluir que la infraestructura de este trabajo corresponde a un Honeypot de alta interacción para ICS debido a sus características. Como desventajas se encuentra que la infraestructura solamente responderá al comportamiento programado por la biblioteca para Modbus de Arduino, lo cual limita el estudio del comportamiento de dispositivos de ICS de diferentes fabricantes que son los que se usan en la industria. En este sentido se ha cumplido el objetivo de crear un entorno de pruebas sencillo de ICS, se han publicado los archivos para que cualquier persona pueda replicar la infraestructura.

Otro de los objetivos logrados fue el encontrar las variables que permiten predecir cuando un flujo es anómalo o no de acuerdo a su valor de *Score*, una de las ventajas de usar el clasificador ingenuo de Bayes es que a diferencia de otros métodos que utilizan procesos más complejos, es fácil identificar el significado de cada variable de acuerdo al contexto en que se esté usando, es decir, se puede interpretar fácilmente porqué una variable $var_j[x_i]$ obtiene tal valor de *Score* con base en la interpretación de los protocolos de comunicación IP, TCP y Modbus en el contexto de Sistemas de Control Industrial. Así pues, el clasificador ingenuo de Bayes es una técnica sencilla que permite con pocos recursos de cómputo obtener resultados buenos, como los discutidos en esta investigación. Para que tenga un mejor desempeño es

necesario el conocimiento de personas expertas en el tema de Sistemas de Control Industrial y de análisis de tráfico de red, ya que su experiencia resulta fundamental para la decisión de incluir o excluir variables durante el entrenamiento, además de poder aplicar adecuadamente los criterios para transformar valores continuos en discretos de las variables de interés, con la técnica de *Coarse grain*. Una desventaja de este método es que si se agrega o elimina alguna de las variables será necesario realizar de nuevo el procedimiento para obtener un nuevo clasificador. Por otro lado, si ataques desconocidos presentan comportamientos similares al modelado podrán ser identificados sin necesidad de realizar un nuevo clasificador. Es posible considerar que las variables $var_j[x_i]$ con valor $P(c \mid x) = 1$ son deterministas ya que se presentaron en 100 % de flujos anómalos y 0 % de flujos normales, con lo que más bien este modelo se asemejaría al de detección de anomalías basadas en firmas, si se sigue esta idea se tendría que crear una firma por cada $var_j[x_i]$ que cumpla con este comportamiento, pero si se llegara a presentar un flujo de tráfico normal con el comportamiento mencionado en una variable, inmediatamente será clasificado como anómalo sin tomar en cuenta los demás parámetros de red, es decir, esa sería la única característica que determinaría que pertenece a tráfico anómalo desde el punto de vista de la detección basada en firmas, sin embargo con el enfoque de la detección basada en anomalías usando el clasificador ingenuo de Bayes, el valor de *Score* de esa variable solamente sumaría al *Score* total del flujo, y si la suma de los *Score* de todas su variables $var_j[x_i]$ es mayor a -9.07 sólo así sería considerado como anómalo, en otro caso sería considerado como tráfico normal, se cree que bajo ciertas condiciones con el clasificador ingenuo de Bayes se disminuiría la tasa de Falsos Positivos para ataques de suplantación en comparación con la detección de intrusos basada en firmas.

Con respecto a la hipótesis de investigación, se comprueba al encontrar un clasificador con métricas aceptables, que permite diferenciar entre el tráfico anómalo del normal para un ambiente de pruebas de ICS que usa como protocolo de comunicación a Modbus, para un ataque de suplantación específico, es importante mencionar que el ambiente de pruebas es sencillo y reducido. Se recomienda seguir con el estudio para disminuir la tasa de Falsos Positivos ya que es más relevante que en un Sistema de Control Industrial las instrucciones de control se lleven a cabo de forma correcta, a diferencia de las redes de servicios convencionales. También será necesario probar este clasificador en otros ambientes de pruebas de ICS para conocer su efectividad.

Como anotación final, se propone que esta metodología presentada en este trabajo se aplique al estudio del tráfico de red de sistemas con servicios convencionales como los que usan los protocolos HTTP, SNMP, ICMP, DNS entre otros, para identificar las anomalías. Desde el punto de vista de la Minería de datos sería un problema de interés debido a que estos sistemas son más abiertos y menos controlados que en un ICS, por lo que se tendría mayor cantidad y variedad de datos a analizar.

8.1. Trabajo Futuro

Durante la investigación surgieron varias ideas de estudio que debido al tiempo y al alcance de este trabajo no se elaboraron, a continuación se enlistan puedan ser estudiadas posteriormente,

- Ampliar la infraestructura de ICS elaborada con diferentes sistemas a controlar como por ejemplo motores, sensores de temperatura y humedad, barómetros, niveles de tanques de agua entre otros, para poder contar con datos más apegados a los de la industria.
- Modificar o elaborar nuevas bibliotecas para Arduino que implementen más funciones del protocolo Modbus.
- Diseñar otro tipo de ataques de la taxonomía descrita para estudiar su comportamiento.
- Analizar y proponer un método de procesamiento más detallado para los datos de la capa de aplicación, es decir los datos de códigos de función, identificador de unidad entre otros, del protocolo Modbus.
- Utilizar otros métodos de detección de anomalías y comparar su efectividad con el clasificador ingenuo de Bayes.
- Seguir la misma metodología presentada en este trabajo pero aplicada al protocolo DNP3 que también es utilizado ampliamente en la industria de ICS.

Bibliografía

Al-Dalky, R., Abduljaleel, O., Salah, K., Otrok, H., and Al-Qutayri, M. (2014). A Modbus traffic generator for evaluating the security of SCADA systems. *2014 9th International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP)*, pages 809–814.

Ask Wireshark (2012). Modbus TCP/IP Problem detectable with WireShark? `https://ask.wireshark.org/questions/15859/modbus-tcpip-problem-detectable-with-wireshark`. Fecha de acceso: noviembre de 2015.

Beckhoff Sist. Industriales (2016). Qué es un PAC. `www.logicelectronic.com/BECKHOFF/QueesunPAC.htm`. Fecha de acceso: septiembre de 2016.

Bhatia, S., Kush, N., Djamaludin, C., Akande, J., and Foo, E. (2014). Practical Modbus flooding attack and detection. *Conferences in Research and Practice in Information Technology Series*, 149:57–65.

Bhattacharyya, D. K. and Kalita, J. K. (2014). *Network Anomaly Detection. A Machine Learning Perspective*. CRC Press.

Bullard, C. (2007). *ra(1), Manual de usuario de Linux. v3.0.8*. Qosient.

Burgueño, M., García-Bastos, J., and González-Buitrago, J. (1995). Las curvas ROC en la evaluación de las pruebas diagnósticas. *Medicina Clínica*, 104(17):661–670.

Cloudshark (2012). Captura de tráfico Modbus. `https://www.cloudshark.org/captures/76038eaa4a3b`. Fecha de acceso: septiembre de 2015.

Combs, R. (2012). Snort 2.9.2: SCADA Preprocessors. `http://blog.snort.org/2012/01/snort-292-scada-preprocessors.html`. Fecha de acceso: septiembre de 2016.

Devarajan, G. (2007). Unraveling scada protocols: Using sullely fuzzer. *Tipping Point DVLabs - DEFCON 15*.

Digital Bond (2011). Quickdraw SCADA IDS. `http://www.digitalbond.com/tools/quickdraw`. Fecha de acceso: marzo de 2016.

Digital Bond (2012). 3S CoDeSys. `http://www.digitalbond.com/tools/basecamp/3s-codesys/`. Fecha de acceso: mayo de 2016.

Fairley, P. (2016). Cybersecurity at U.S. utilities due for an upgrade: Tech to detect intrusions into industrial control systems will be mandatory. *IEEE Spectrum*, 53(May, 2016):11–13.

Fall, K. R. and Stevens, W. R. (2012). *TCP/IP Illustrated. Volume 1. The Protocols.* Addison-Wesley.

FireEye (2016). 2016 ICS Vulnerability Trend Report. Technical report.

FleaPLC (2014). Arduino as Raspberry PI's remote IO (Codesys) . `http://www.fleaplc.it/en/tutorials/33-arduino-as-raspberry-pi-s-remote-io-codesys`. Fecha de acceso: abril de 2016.

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: techniques, systems and challenges. *computers & security 28*.

Goldenberg, N. and Wool, A. (2013). Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection*, 6(2):63–75.

Hjelmvik, E. (2015). Capture files from 4SICS Geek Lounge. `https://www.netresec.com/?page=PCAP4SICS`. Fecha de acceso: febrero de 2016.

Huitsing, P., Chandia, R., Papa, M., and Shenoi, S. (2008). Attack taxonomies for the Modbus protocols. *International Journal of Critical Infrastructure Protection*, 1(C):37–44.

Internet Assigned Numbers Authority (2016). Service Name and Transport Protocol Port Number Registry. Consultada en enero de 2016.

Jiménez Díaz, J. (2011). *Using SNORT for intrusion detection in MODBUS TCP/IP communications.* InfoSec Reading Room. SANS Institute.

Knapp, E. D. and Langill, J. T. (2015). *Industrial Network Security. Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems.* Number 2. Syngress, Elsevier Inc.

López Barrientos, M. J. and Quezada Reyes, C. (2006). *Fundamentos de seguridad informática.* UNAM, Fac. de Ingeniería.

McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A. R., Maniatakos, M., and Karri, R. (2016). The Cybersecurity Landscape in Industrial Control Systems. *Proceedings of the IEEE*, 104(5):1039–1057.

Modbus Organization (2002). MODBUS over serial line–Specification and Implementation guide.

Modbus Organization (2006). Modbus Messaging on TCP/IP Implementation Guide V1.0b.

Modbus Organization (2009). Conformance test specification for Modbus TCP v3.

Morris, T. H., Jones, B. A., Vaughn, R. B., and Dandass, Y. S. (2013). Deterministic intrusion detection rules for MODBUS protocols. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 1773–1781.

National Instruments (2014). The Modbus Protocol In-Depth. Consultada en enero de 2016.

NERC (2016). North American Electric Realiability Corporation NERC. `http://www.nerc.com`. Fecha de acceso: septiembre de 2016.

Northcutt, S. and Novak, J. (2002). *Network Intrusion Detection*. New Riders Publishing.

Obregon, L. (2015). Secure Architecture for Industrial Control Systems. *SANS Institute*, page 26.

Organization, M. (2012). *Modbus applicaction protocol specification v1.1b3*.

Pretorius, B. and van Niekerk, B. (2015). Cyber-Security and Governance for ICS/SCADA in South Africa. In *Iccws 2015-The Proceedings of the 10th International Conference on Cyber Warfare and Security: ICCWS2015*, page 241. Academic Conferences Limited.

RFC791 (1981). Internet Protocol.

RFC793 (1981). Transmission Control Protocol.

Santillan, J. (2014). Capita Selecta Report. Data sequences analysis for anomaly and intrusion detection on ICS network protocols. *Technology University Eindhoven*.

Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M., and Hahn, A. (2015). NIST Special Publication 800-82: Guide to Industrial Control Systems (ICS) Security.

Tanenbaum, A. S. and Wetherall, D. J. (2012). *Redes de computadoras*. Prentice Hall.

Valencia, A. I. (2017). Detección y clasificación de Malware: Un enfoque basado en algoritmos de Aprendizaje Supervisado. Master's thesis, Universidad Nacional Autónoma de México. Posgrado en Ciencia e Ingeniería de la Computación.

Witte Software (2015). Modbus tools. `http://www.modbustools.com`. Fecha de acceso: diciembre de 2015.

Zhu, B., Joseph, A., and Sastry, S. (2011). A Taxonomy of Cyber Attacks on SCADA Systems. pages 380–388.

Apéndice A

Glosario

Análisis de tráfico de red. Es una técnica que consiste en obtener información a partir del procesamiento del tráfico de una red. Esta técnica permite obtener información como rendimiento de la red, consumo de recursos, utilización del ancho de banda, protocolos de red utilizados, y detectar comportamientos anómalos.

Analizador de protocolos. Herramienta que permite analizar los diferentes protocolos del modelo TCP/IP en el tráfico de red, muestra información hexadecimal de los paquetes y la gran mayoría realiza una interpretación de los mismos. Como ejemplos se tiene Wireshark, Tcpdump, Ettercap y Kismet. Son comúnmente conocidos como sniffer.

Se requieren permisos administrativos en el sistema operativo para su ejecución porque colocan la interfaz de red en modo monitor, esto con el fin de que se procesen todos los paquetes de red incluyendo aquellos que no son destinados para la dirección IP asociada a la interfaz de red.

Coil. En el protocolo Modbus es un valor binario, por ejemplo encendido o apagado (*on* u *off*).

Detección de anomalías. Los métodos de detección de anomalías están basados en el análisis del comportamiento del tráfico normal de tal forma que perfiles generados a partir de una línea base (baseline), en base a las actividades del tráfico legítimo, son definidos para identificar actividad potencialmente maliciosa. La principal característica de este tipo de modelo de detección es que dan la capacidad de identificar ataques imprevistos.

Digital Bond. Asociación creada en 1998, que realiza evaluaciones, conferencias, buenas prácticas y consultorías sobre la seguridad en ICS.

IDS e IPS. Intrusion Detection System e Intrusion Prevention System respectivamente, son sistemas físicos o lógicos que analizan el tráfico de red con el fin de encontrar anomalías. Su funcionamiento está basado en la detección de tráfico malicioso a través de firmas o de la identificación de comportamiento anómalo.

Los IDS realizan el análisis de tráfico de forma pasiva, es decir, solamente alertan sobre una posible anomalía; mientras que los IPS pueden llegar a tomar alguna acción a partir de la detección de la anomalía, como por ejemplo reducir el ancho de banda para ese dispositivo, bloquear completamente el tráfico anómalo o simplemente alertar.

Machine Learning. El aprendizaje automático, es una rama de la inteligencia artificial, en la que a partir del modelado de datos con diferentes técnicas, es posible predecir o clasificar dicho datos y obtener un comportamiento esperado. Es decir, generalizar comportamientos a partir de información no estructurada. Comúnmente se divide en aprendizaje supervisado y no supervisado.

Modelo OSI. Open System Interconnection, es un modelo de referencia para la interconexión de sistemas de comunicaciones, creado en 1980 por la Organización Internacional de Estándares (International Organization for Standardization, ISO) y consta de siete capas.

Modelo TCP/IP. Modelo usado para la interconexión de dispositivos de comunicaciones a través de la implementación de diversos protocolos de red. Propuesto en los años 1970, consta de cuatro capas.

Proceso industrial. La parte principal del sistema, es decir, el objetivo específico de una industria, es conocido como el proceso industrial.

PDU. Protocol Data Unit, unidad de datos de un protocolo.

Transacción de Modbus. Se conoce así al intercambio de información entre dispositivos que utilizan el protocolo Modbus. Una transacción Modbus consiste en un par de paquetes de solicitud o *Request* y de respuesta o *Response*, ambos con el mismo Identificador de transacción.

Register. En el protocolo Modbus corresponde a una valor de lectura análogo, por ejemplo de la Y a la Z

Apéndice B

Interpretación de los datos de los PLC

B.1. PLC para un fotosensor

El dispositivo esclavo, un PLC construido con Arduino, recibe el valor de luminosidad de una fotoresistencia de un circuito eléctrico, y dependiendo del valor de ésta envía valores de voltaje al circuito para encender uno, dos o tres led, mientras mayor sea el valor de luminosidad mayor el número de led encendidos.

B.1.1. Función *Write Multiple Coils.*

A cada led se le asigna una salida discreta o *Coil*, un valor de 1 significa que el led debe encender, un valor de 0 significa que el led debe permanecer apagado, es decir, son valores de un bit por cada salida discreta, un bit por cada led. Si se requiere encender el 1er y 2o led, se tendrá que enviar la secuencia de bits 011, se usan tres bits puesto que son tres led. Es necesario completar o rellenar con bits para tener un byte, entonces la secuencia de bits es 00000011, y en formato hexadecimal es igual a 0x03, estos datos son los que el maestro deberá a enviar al esclavo, previo establecimiento de la conexión, para encender dichos led. En la figura B-1 se muestra la configuración del PLC utilizado así como los datos que puede enviar el maestro al esclavo, dichos datos llegarán al esclavo, es decir, el maestro envía datos para guardar o escribir en la memoria del esclavo, de esta forma el esclavo enviará estos datos a través de sus salidas hacia el circuito eléctrico.

Datos en Modbus PDU	Interpretación
0x00	led apagados
0x01	1er led encendido
0x03	1er y 2o led encendidos
0x07	1er, 2o y 3er led encendidos

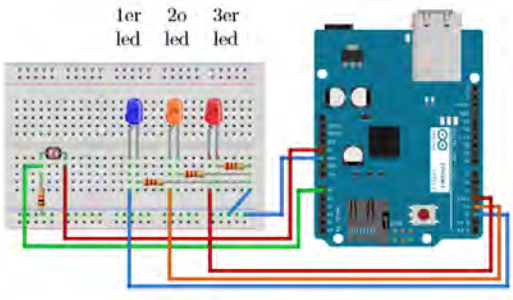


Figura B-1: Datos en Modbus PDU para el PLC del fotosensor.
Función *Write Multiple Coils.*

El maestro deberá enviar un paquete *Request* al esclavo para indicarle a través de los datos de Modbus PDU, como es que se deben de controlar los led. A continuación se muestra un ejemplo de una transacción Modbus, en el cual el maestro le indica al esclavo que los led deben permanecer apagados, para esto el valor a enviar será de 0 para cada uno de ellos, por lo que la secuencia completa de bits a enviar al esclavo será 00000000, y en formato hexadecimal es igual a 0x00. Los datos marcados con color verde corresponden al protocolo IP, con color azul los que corresponden al protocolo TCP, con color morado los que corresponden a la cabecera Modbus y con color amarillo los que corresponden al PDU de Modbus.

```
18:12:44.107831 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [P.], seq 1:15, ack 1,
win 29200, length 14
        0x0000: 4500 0036 adfe 4000 4006 08b6 c0a8 0145 E..6..@.@......E
        0x0010: c0a8 0178 9c85 01f6 164e 1ffd f4d6 47d2 ...x.....N....G.
        0x0020: 5018 7210 8436 0000 0002 0000 0008 020f P.r..6..........
        0x0030: 0000 0008 0100                          ......
```

Cabecera de Modbus (MBAP Header):
- Transaction identifier: 0x0002
- Protocol Identifier: 0x0000
- Length: 0x0008
- Unit Identifier: 0x02

Modbus PDU:
- Function Code: 0x0f, Write Multiple Coils (15)
- Starting Address: 0x0000
- Quantity of outputs: 0x0008
- Byte Count: 0x01
- Output value: 0x00

En este paquete el maestro envía un *Request* con un identificador de transacción o *Transaction identifier* igual a 2, por lo que se espera que en el paquete *Response* el identificador de transacción corresponda al recibido, es decir al mismo 2. El identificador de protocolo o *Protocol identifier*, corresponde a 0x0000, el valor para Modbus. El campo *Length* corresponde a 8 bytes, es el número de bytes que restan del paquete, incluyendo el identificador de unidad y la PDU. Para este caso el maestro a asignado como identificador de unidad o *Unit identifier* al esclavo con el valor de 2.

El código de función es el 15, el cual corresponde a la función *Write Multiple Coils*, esto significa que el maestro le manda una instrucción al esclavo de escritura a la memoria. El acceso a la memoria del esclavo será desde el inicio, lo indica con el valor 0x0000 del campo *Starting Address*. La cantidad de salidas o *Quantity of outputs* será de 8 bits, que corresponden a los tres bits para los led más cinco bits de relleno. La cantidad de bytes o *Byte Count* a usar está en función de la cantidad de salidas y se calcula como

$$N = \textit{Cantidad de salidas}/8,$$

para este caso

$$N = 8/8 = 1 \text{ byte.}$$

Los valores de salida o *Outputs value* corresponden a la forma en cómo se requieren encender o apagar los led, como ya se explicó antes, para este caso su valor es 0x00. Con estos valores es como el maestro construye el PDU del paquete *Request*.

Después del paquete *Request* por parte del maestro, el esclavo envía un paquete *Response* en respuesta al primero, a continuación se muestra el paquete *Response* del esclavo.

```
18:12:44.129793 IP 192.168.1.120.502 >192.168.1.69.40069: Flags [P.], seq 1:13, ack 15,
win 2048, length 12
       0x0000: 4500 0034 0ec2 4000 8006 67f4 c0a8 0178 E..6..@.@.....E
       0x0010: c0a8 0145 01f6 9c85 f4d6 47d2 164e 200b ...x.....N....G.
       0x0020: 5018 0800 1016 0000 0002 0000 0006 020f P.r..6.........
       0x0030: 0000 0008                               ......
```

Cabecera de Modbus (MBAP Header):
- Transaction identifier: 0x0002
- Protocol Identifier: 0x0000
- Length: 0x0006
- Unit Identifier: 0x02

Modbus PDU:
- Function Code: 0x0f, Write Multiple Coils (15)
- Starting Address: 0x0000
- Quantity of outputs: 0x0008

El identificador de transacción corresponderá al mismo que el enviado por el maestro, de esta forma el protocolo Modbus establece un control entre solicitudes y respuestas. El identificador de protocolo corresponde al 0. El valor del campo *Length* es de 6 bytes, que es el número de bytes que restan del paquete, incluyendo el identificador de unidad y la PDU. El identificador de unidad debe permanecer igual que el enviado en el *Request*.

El código de función debe ser el mismo que en el *Request*, que corresponde a la función *Write multiple coils*. El valor inicial de memoria es 0x0000 que fue al que se accedió y corresponde al mismo que en el *Request* del maestro. El valor de salidas corresponde al número que se escribieron en la memoria del esclavo, que corresponde a 8 bits, 3 para los led más 5 para completar un byte. En este mismo paquete se envía adicionalmente de la bandera PUSH de TCP, la bandera ACK para indicar que el paquete anterior de TCP fue recibido satisfactoriamente, es decir, el enviado por el maestro.

En el siguiente paquete se observa la confirmación del protocolo TCP con la bandera ACK activada, por parte del maestro de la llegada del paquete *Response* del esclavo.

```
18:12:44.130077 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [.], ack 13, win 29200,
length 0
       0x0000: 4500 0028 adff 4000 4006 08c3 c0a8 0145 E..(..@.@......E
       0x0010: c0a8 0178 9c85 01f6 164e 200b f4d6 47de ...x.....N....G..
       0x0020: 5010 7210 8428 0000                     P.r..(..
```

B.1.2. Función *Read Holding Registers.*

El maestro le envía al esclavo una solicitud de lectura de sus registros internos o *Holding registers*, estos valores son leídos por el PLC del fotosensor conectado, es decir de una lectura análoga. El PLC realiza una transformación del valor recibido por el fotosensor y lo envía al maestro, previa solicitud de por medio. Dependiendo del rango en que se encuentre el valor enviado por el PLC, el maestro le enviará una solicitud para encender uno, dos o los tres led del circuito. Los registros a los que accede el maestro son de 16 bits, puede ser de 1 a 125 registros [Organization, 2012]. En la figura B-2 se muestra la configuración del PLC utilizado así como los datos que puede recibir el maestro del esclavo.

Datos en Modbus PDU (de la lectura del fotosensor)	Acción a enviar al esclavo
< 256	led apagados
256 - 512	encender 1er led
512 - 768	encender 2o led
> 768	encender 1er, 2o y 3er led

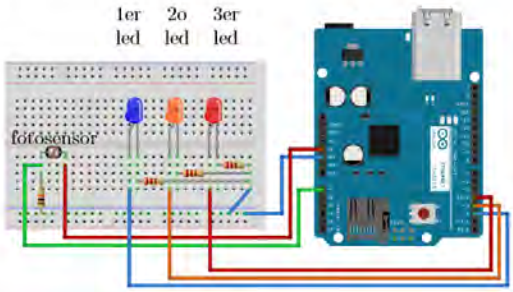


Figura B-2: Datos en Modbus PDU para el PLC del fotosensor.
Función *Read Holding Registers*

El maestro deberá enviar un paquete *Request* al esclavo para solicitarle el valor de luminosidad que obtuvo del fotosensor. A continuación se muestra este paquete.

```
18:12:44.147464 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [P.], seq 15:27, ack 13,
win 29200, length 12
        0x0000: 4500 0034 ae00 4000 4006 08b6 c0a8 0145 E..4..@.@......E
        0x0010: c0a8 0178 9c85 01f6 164e 200b f4d6 47de ...x.....N....G.
        0x0020: 5018 7210 8434 0000 0003 0000 0006 0203 P.r..4..........
        0x0030: 0000 0008                               ....
```

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x0003
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0x02

Modbus PDU:
Function Code: 0x03, Read Holding Registers (3)
Starting Address: 0x0000
Quantity of registers: 0x0008

En este paquete el maestro envía un *Request* con un identificador de transacción o *Transaction identifier* igual a 3, por lo que se espera que en el paquete *Response* el identificador de transacción corresponda al mismo. El identificador de protocolo o *Protocol identifier*, corresponde a 0x0000, el valor para Modbus. El valor de 6 bytes corresponde al campo *Length*, estos bytes son los que restan del paquete, incluyendo el identificador de unidad y la PDU. Para este caso el maestro a asignado como identificador de unidad o *Unit identifier* al esclavo con el valor de 2.

El código de función es el 3, el cual corresponde a la función *Read Holding Registers*, esto significa que el maestro le manda una instrucción al esclavo de lectura de sus registros internos de 16 bits. El acceso a la memoria del esclavo será desde el inicio, lo indica con el valor 0x0000 del campo *Starting Address*. La cantidad de registros o *Quantity of registers* a leer, para este caso será de 8.

Después del paquete *Request* por parte del maestro, el esclavo envía un paquete *Response* en respuesta al primero, a continuación se muestra el paquete *Response* del esclavo.

```
18:12:44.167026 IP 192.168.1.120.502 >192.168.1.69.40069: Flags [P.], seq 13:38, ack 27, win
2048, length 25
        0x0000: 4500 0041 0ec3 4000 8006 67e6 c0a8 0178 E..A..@...g....x
        0x0010: c0a8 0145 01f6 9c85 f4d6 47de 164e 2017 ...E......G..N..
        0x0020: 5018 0800 9af6 0000 0003 0000 0013 0203 P...............
        0x0030: 1000 6500 0000 0000 0000 0000 0000 0000 ..e.............
        0x0040: 00                                      .
```

Cabecera de Modbus (MBAP Header):
- Identificador de transacción: 0x0003
- Protocol Identifier: 0x0000
- Length: 0x0013
- Unit Identifier: 0x02

Modbus PDU:
- Function Code: 0x03, Read Holding Registers (3)
- Byte Count: 0x0010
- Register 0 (UINT16): 0x0065
- Register 1 (UINT16): 0x0000
- Register 2 (UINT16): 0x0000
- Register 3 (UINT16): 0x0000
- Register 4 (UINT16): 0x0000
- Register 5 (UINT16): 0x0000
- Register 6 (UINT16): 0x0000
- Register 7 (UINT16): 0x0000

El identificador de transacción corresponderá al mismo que el enviado por el maestro, de esta forma el protocolo Modbus establece un control entre solicitudes y respuestas. El identificador de protocolo corresponde al 0. El valor que corresponde al campo *Length* es de 19 bytes, los bytes restantes del paquete. El identificador de unidad debe permanecer igual que el enviado en el *Request*.

El código de función debe ser el mismo que en el *Request*, que corresponde a la función *Read Holding Registers*. La cantidad de bytes o *Byte Count* a usar está en función de la cantidad de registros y se calcula como

$$Byte\ count = 2 \times N, \text{ en donde } N = cantidad\ de\ registros$$

y para este caso,

$$Byte\ count = 2 \times N = 2 \times 8 = 16 \text{ bytes.}$$

El valor de N se obtuvo del *Request* enviado por el maestro en su campo cantidad de registros o *Quantity of registers*. Entonces, la cantidad de bytes o *Bytes count* es de 16 bytes, y en su representación hexadecimal es 0x10, esto es porque en total son 8 registros los solicitados por el maestro, y cada registro es de 16 bits o 2 bytes. Después se muestran los datos de los registros, para este ejemplo solamente es necesario usar un registro, que será el registro 0, el cual contiene el valor de la luminosidad, en este caso 0x0065, que es 101 en decimal, y de acuerdo a los rangos de luminosidad definidos por el maestro, en una transacción posterior, éste tendrá que enviarle la instrucción al esclavo de que todos los led deben permanecer apagados. En este mismo paquete se envía adicionalmente de la bandera PUSH de TCP, la bandera ACK para indicar que el paquete anterior de TCP fue recibido satisfactoriamente, es decir, el enviado por el maestro.

En el siguiente paquete se observa la confirmación del protocolo TCP con la bandera ACK activada, por parte del maestro de la llegada del paquete *Response* del esclavo.

```
18:12:44.205813 IP 192.168.1.69.40069 >192.168.1.120.502: Flags [.], ack 38, win 29200,
length 0
        0x0000: 4500 0028 ae01 4000 4006 08c1 c0a8 0145 E..(..@.@......E
        0x0010: c0a8 0178 9c85 01f6 164e 2017 f4d6 47f7 ...x.....N....G.
        0x0020: 5010 7210 8428 0000                     P.r..(..
```

B.2. PLC para un semáforo

Este dispositivo PLC esclavo está construido con Arduino, tiene conectado un circuito con dos grupos de tres led de color rojo, amarillo y verde por cada grupo, es decir, es un semáforo. El esclavo recibe del maestro las instrucciones de cómo encender o apagar los led, es decir, dependiendo del valor que envíe el maestro, el esclavo envía valores de voltaje al circuito para encender o apagar los led.

B.2.1. Función *Write Multiple Coils.*

A cada led se le asigna una salida discreta o *Coil*, un valor de 1 significa que el led debe encender, un valor de 0 significa que el led debe permanecer apagado, es decir, son valores de un bit por cada salida discreta, un bit por cada led. En la figura B-3 se muestra la configuración del PLC utilizado así como los datos que puede enviar el maestro al esclavo, dichos datos llegarán al esclavo, es decir, el maestro envía datos para guardar o escribir en la memoria del esclavo, de esta forma el esclavo enviará estos datos a través de sus salidas hacia el circuito eléctrico. Si se requiere encender los led rojo1 y verde2, y que permanezcan apagados el resto de los led, se tendrá que enviar la secuencia de bits 100001, se usan seis bits puesto que son seis led en total, sin embargo es necesario completar o rellenar con bits para tener un byte, entonces se añaden dos bits a la parte más significativa del byte, por lo que la secuencia de bits resultante es 00100001, y su representación en hexadecimal es igual a 0x21, estos datos son los que el maestro deberá enviar al esclavo, previo establecimiento de la conexión, para encender dichos led y dejar apagados el resto.

Datos en Modbus PDU	Interpretación
0x09	encender rojo1 – rojo2
0x21	encender rojo1 – verde2
0x11	encender rojo1 – amarillo2
0x0c	encender verde1 – rojo2
0x0a	encender amarillo1 – rojo2

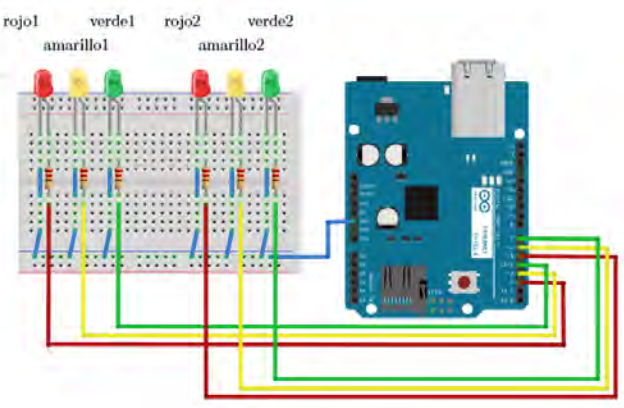


Figura B-3: Significado de los datos en Modbus PDU para el PLC del semaforo. Función *Write Multiple Coils.*

Entonces el maestro deberá enviar un paquete *Request* al esclavo como el que se muestra a continuación.

```
18:12:44.107339 IP 192.168.1.69.58930 >192.168.1.110.502: Flags [P.], seq 1:15, ack 1,
win 29200, length 14
        0x0000: 4500 0036 e9ad 4000 4006 cd10 c0a8 0145 E..6..@.@......E
        0x0010: c0a8 016e e632 01f6 cbeb 5bac 6e91 a08d ...n.2....[.n...
        0x0020: 5018 7210 842c 0000 0001 0000 0008 010f P.r..,..........
        0x0030: 0000 0008 0121                          .....!
```

Cabecera de Modbus (MBAP Header):
- Transaction identifier: 0x0001
- Protocol Identifier: 0x0000
- Length: 0x0008
- Unit Identifier: 0x01

Modbus PDU:
- Function Code: 0x0f, Write Multiple Coils (15)
- Starting Address: 0x0000
- Quantity of outputs: 0x0008
- Byte Count: 0x01
- Outputs value: 0x21

En este paquete el maestro envía un *Request* con un identificador de transacción o *Transaction identifier* igual a 1, por lo que se espera que en el paquete *Response* el identificador de transacción corresponda al mismo 2. El identificador de protocolo o *Protocol identifier*, corresponde a 0x0000, el valor para Modbus. Para el campo *Length* se tiene un valor de 8 bytes, es el número de bytes que restan, incluyendo el identificador de unidad y la PDU. Para este caso el maestro a asignado como identificador de unidad o *Unit identifier* al esclavo con el valor de 1.

El código de función es el 15, el cual corresponde a la función *Write Multiple Coils*, esto significa que el maestro le manda una instrucción al esclavo de escritura a la memoria. El acceso a la memoria del esclavo será desde el inicio, lo indica con el valor 0x0000 del campo *Starting Address*. La cantidad de salidas o *Quantity of outputs* será de 8 bits, que corresponden a los seis bits para los led más dos bits de relleno. La cantidad de bytes o *Byte Count* a usar está en función de la cantidad de salidas y se calcula como

$$N = \textit{Cantidad de salidas}/8,$$

para este caso

$$N = 8/8 = 1 \text{ byte.}$$

Los valores de salida o *Outputs value* corresponden a la forma en cómo se requieren encender o apagar los led, como ya se explicó antes, para este caso su valor es 0x21. Con estos valores es como el maestro construye el PDU del paquete *Request*.

Después del paquete *Request* por parte del maestro, el esclavo envía un paquete *Response* en respuesta al primero, a continuación se muestra el paquete *Response* del esclavo.

```
18:12:44.128799 IP 192.168.1.110.502 >192.168.1.69.58930: Flags [P.], seq 1:13, ack 15,
win 2048, length 12
        0x0000: 4500 0034 0274 4000 8006 744c c0a8 016e E..4.t@...tL...n
        0x0010: c0a8 0145 01f6 e632 6e91 a08d cbeb 5bba ...E...2n.....[.
        0x0020: 5018 0800 03b1 0000 0001 0000 0006 010f P...............
        0x0030: 0000 0008                               ....
```

Cabecera de Modbus (MBAP Header):
Transaction identifier: 0x0001
Protocol Identifier: 0x0000
Length: 0x0006
Unit Identifier: 0x01

Modbus PDU:
Function Code: 0x0f, Write Multiple Coils (15)
Starting Address: 0x0000
Quantity of outputs: 0x0008

El identificador de transacción corresponderá al mismo que el enviado por el maestro, de esta forma el protocolo Modbus establece un control entre solicitudes y respuestas. El identificador de protocolo corresponde al 0. El valor para el campo *Length* es de 6 bytes, este valor corresponde a los bytes restantes del paquete. El identificador de unidad debe permanecer igual que el enviado en el *Request*.

El código de función debe ser el mismo que en el *Request*, que corresponde a la función *Write multiple coils*. El valor inicial de memoria es 0x0000 que fue al que se accedió y corresponde al mismo que en el *Request* del maestro. El valor de salidas corresponde al número que se escribieron en la memoria del esclavo, que corresponde a 8 bits, seis para los led más dos para completar un byte. En este mismo paquete se envía adicionalmente de la bandera PUSH de TCP, la bandera ACK para indicar que el paquete anterior de TCP fue recibido satisfactoriamente, es decir, el enviado por el maestro.

En el siguiente paquete se observa la confirmación del protocolo TCP con la bandera ACK activada, por parte del maestro de la llegada del paquete *Response* del esclavo.

```
18:12:44.129238 IP 192.168.1.69.58930 >192.168.1.110.502: Flags [.], ack 13, win 29200,
length 0
       0x0000: 4500 0028 e9ae 4000 4006 cd1d c0a8 0145 E..(..@.@......E
       0x0010: c0a8 016e e632 01f6 cbeb 5bba 6e91 a099 ...n.2....[.n...
       0x0020: 5010 7210 841e 0000                     P.r.....
```

Apéndice C

Circuitos

Diagramas de los circuitos electrónicos para el sensor de luminosidad y para el semáforo[1].

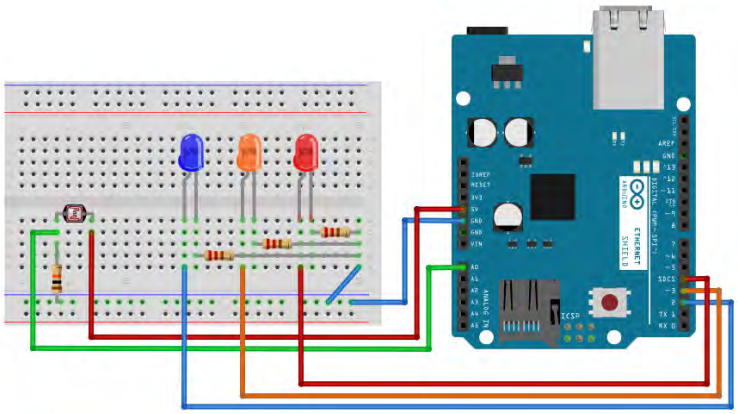


Figura C-1: Diagrama de conexión del circuito fotosensor

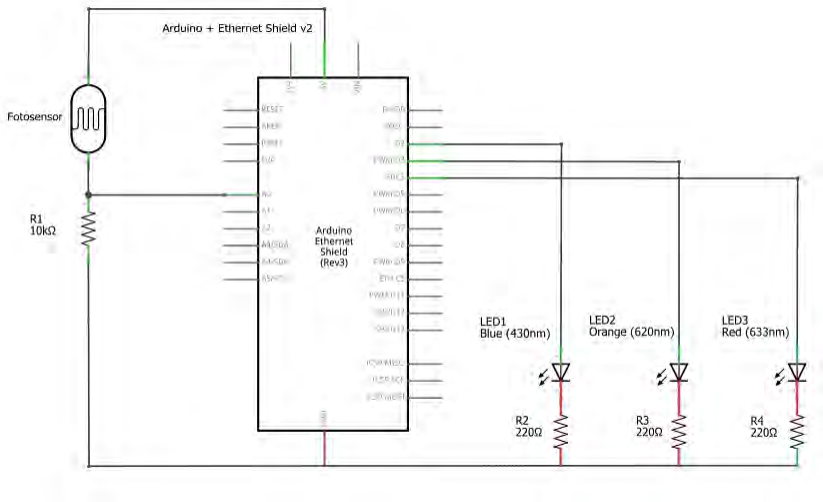


Figura C-2: Esquema eléctrico de conexión del circuito fotosensor

[1]Se realizaron usando el software Fritzing, `http://fritzing.org`.

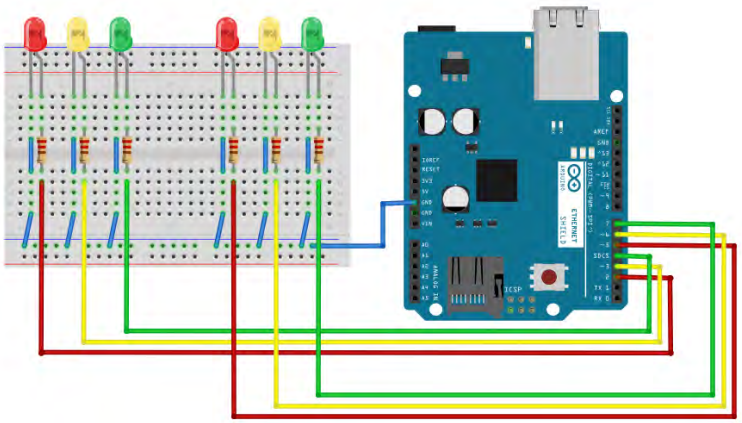


Figura C-3: Diagrama de conexión del circuito semáforo

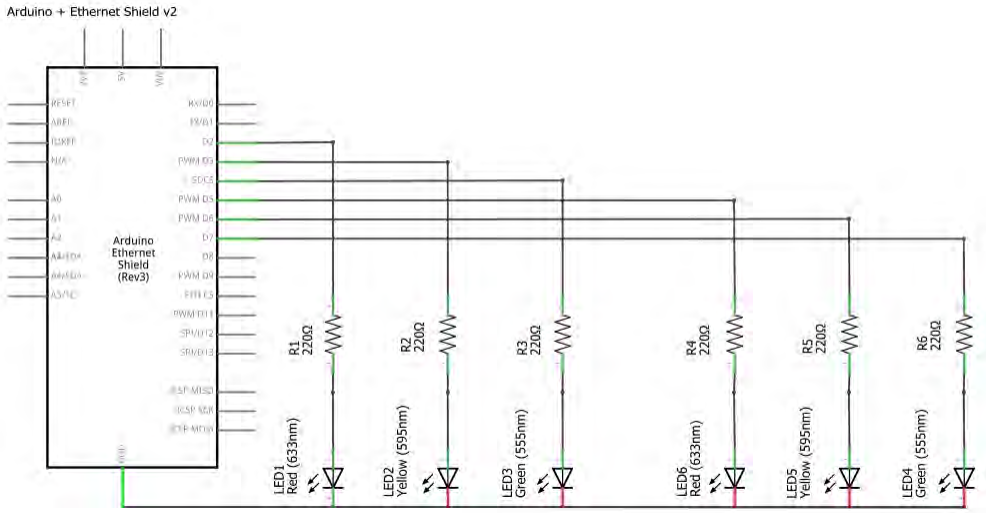


Figura C-4: Esquema eléctrico de conexión del circuito semáforo

Apéndice D

Diccionario de datos

D.1. Variables utilizadas

Campo	Significado	Descripción
Flgs	*Flags*	Banderas del estado del flujo observadas en la transacción. Corresponden a identificadores de protocolo, estados o atributos, de los flujos.
Dur	*Duration*	Duración total del flujo, en segundos. Se obtiene de la diferencia del tiempo de término (campo *ltime*) y el tiempo de inicio (campo *stime*) del flujo, es decir: $ltime - stime$.
SrcAddr	*Source Address*	Dirección IP origen, se considera origen al host que transmite primero un paquete como parte de un flujo.
DstAddr	*Destination Address*	Dirección IP de destino.
Proto	*Protocol*	Protocolo de la transacción.
Sport	*Source port*	Número de puerto origen.
Dport	*Destination port*	Número de puerto de destino.
sTos	*source TOS*	Valor del byte del campo Tipo de servicio (Type of Service) de la cabecera IP, para el origen.
dTos	*destination TOS*	Valor del byte del campo Tipo de servicio (Type of Service) de la cabecera IP, para el destino.
sDSb	*source Differentiated Service Byte*	Valor del byte del campo Servicios diferenciados (Differentiated Service) de la cabecera IP, para el origen.
dDSb	*destination Differentiated Service Byte*	Valor del byte del campo Servicios diferenciados (Differentiated Service) de la cabecera IP, para el destino.
sTtl	*source TTL*	Valor del campo de Tiempo de vida (Time to Live) de la cabecera IP, para el flujo del origen al destino.
dTtl	*destination TTL*	Valor del campo de Tiempo de vida (Time to Live) de la cabecera IP, para el flujo del destino al origen.
sHops	*source Hops*	Número de saltos (basado en direcciones IP) estimados desde el origen al punto en que se tomó la captura de tráfico.
dHops	*destintation Hops*	Número de saltos (basado en direcciones IP) estimados desde el destino al punto en que se tomó la captura de tráfico.
sIpId	*source IP Identifier*	Número de Identificación de la cabecera IP, para el origen.
dIpId	*destination IP Identifier*	Número de Identificación de la cabecera IP, para el destino.
Cause		Código de causa de Argus.
SrcPkts	*Source Packets*	Número de paquetes del flujo del origen al destino.
DstPkts	*Destination Packets*	Número de paquetes del flujo del destino al origen.
SrcBytes	*Source Bytes*	Bytes de la transacción, del origen al destino.
DstBytes	*Destination Bytes*	Bytes de la transacción, del destino al origen.
PCRatio	*Producer Consumer Ratio*	Es un valor normalizado que indica la dirección de la transferencia de información de la capa de aplicación. Se calcula con la fórmula: $PCR = \frac{SrcApplicationBytes - DstApplicationBytes}{SrcApplicationBytes + DstApplicationBytes}$

Tabla D.1: Variables utilizadas en el análisis (1)

Campo	**Significado**	**Descripción**
SrcLoad	*Source Load*	Bits por segundo del origen
DstLoad	*Destination Load*	Bits por segundo del destino
SrcLoss	*Source Loss*	Paquetes del origen retransmitidos o descartados.
DstLoss	*Destination Loss*	Paquetes del destino retransmitidos o descartados.
SrcRetra	*Source Retransmitted*	Paquetes del origen retransmitidos.
DstRetra	*Destination Retransmitted*	Paquetes del destino retransmitidos.
SrcGap	*Source Gap*	Bytes del origen perdidos en el flujo de datos.
DstGap	*Destination Gap*	Bytes del destino perdidos en el flujo de datos.
SIntPkt	*Source InterPacket*	Tiempo de llegada entre paquetes del origen, en milisegundos.
DstIntPkt	*Destination InterPacket*	Tiempo de llegada entre paquetes del destino, en milisegundos.
SrcJitter	*Source Jitter*	Variación en el retardo entre paquetes del mismo flujo del origen, en milisegundos.
DstJitter	*Destination Jitter*	Variación en el retardo entre paquetes del mismo flujo del origen, en milisegundos.
State		Reporta el principal estado de la transacción, y depende del protocolo. Para todos los protocolos, excepto ICMP, este campo informa sobre el estado básico de una transacción.
SrcWin	*Source Window*	Notificación del tamaño de ventana del origen. Con las notifiaciones del tamaño de ventana el destino informa continuamente al origen cuantos datos (en bytes) está preparado para recibir.
DstWin	*Destination Window*	Notificación del tamaño de ventana del destino. Con las notificaciones del tamaño de ventana el destino informa continuamente al origen cuantos,datos (en bytes) está preparado para recibir.
Anomaly		Indica si el flujo es anómalo o no.

Tabla D.2: Variables utilizadas en el análisis (2)

D.2. Variables descartadas.

Campo	Significado	Descripción
Seq	*Sequence number*	Número de secuencia que es asignado por Argus; depende del tráfico procesado para capturas de tráfico diferentes el número se puede repetir, es decir, se reasignan por cada nueva captura.
SAppBytes	*Source Application Bytes*	Bytes de la capa de aplicación del destino al origen. Se calcula como: $Appbytes = (Bytes\,totales - \sum_{i=2}^{4} Bytes\,cabecera\,capa_i) - Bytes\,retransmitidos$ Se usó el campo PCRatio que relaciona a los campos SAppBytes y DAppBytes.
DAppBytes	*Destination Application Bytes*	Bytes de la capa de aplicación del destino al origen. Se calcula como: $Appbytes = (Bytes\,totales - \sum_{i=2}^{4} Bytes\,cabecera\,capa_i) - Bytes\,retransmitidos$ Se usó el campo PCRatio que relaciona a los campos SAppBytes y DAppBytes.
SrcRate	*Source Rate*	Número de paquetes por segundo transmitidos por el origen. Se calcula con pkts/(ltime-stime), por lo tanto es dependiente de las variables SrcPkts y Dur.
DstRate	*Destination Rate*	Número de paquetes por segundo transmitidos por el destino. Se calcula con pkts/(ltime-stime), por lo tanto es dependiente de las variables DstPkts y Dur.
Dir	*Direction*	Dirección de la transacción, se usa para indicar cuales host están transmitiendo. Para TCP este campo indica el origen de la conexión TCP, y el caracter del centro indica el estado de la transacción. Es necesario capturar el inicio de la conexión para contar con la dirección real del flujo.
SynAck	*Synchronization-Acknowledgement*	Tiempo de establecimiento de la conexión TCP. Es el tiempo entre los paquetes de SYN y de SYN_ACK. En la captura de tráfico pueden no estar presente el inicio de la conexión TCP, es decir, el Three Way Handshake.
AckDat	*Acknowledgement-Data*	Tiempo de establecimiento de la conexión TCP. Es el tiempo entre los paquetes de SYN_ACK y ACK. En la captura de tráfico puede no estar presente el inicio de la conexión TCP, es decir, el Three Way Handshake.
TcpOpt	*TCP Options*	Opciones del protocolo TCP registradas al inicio de una conexión. En la captura de tráfico puede no estar presente el inicio de la conexión TCP.
sMeanPktSz	*source Mean Packet Size*	Media del tamaño del paquete del flujo transmitida por el origen. Dependiente de tamaño del paquete (variables SrcBytes y SAppBytes).
dMeanPktSz	*destination Mean Packet Size*	Media del tamaño del paquete del flujo transmitida por el destino. Dependiente de tamaño del paquete (variables DstBytes y DAppBytes).
SrcTCPBase	*Source TCP Base*	Número de secuencia base para el protocolo TCP, del origen. Número pseudoaletario generado por el sistema operativo, cambia con cada nueva conexión, tanto en las anómalas como en las no anómalas.
DstTCPBase	*Destination TCP Base*	Número de secuencia base para el protocolo TCP, del destino. Número pseudoaletario generado por el sistema operativo, cambia con cada nueva conexión, tanto en las anómalas como en las no anómalas.
Offset		Desplazamiento de bytes en el archivo o en el flujo de datos.

Tabla D.3: Variables descartadas para el análisis

D.3. Valores utilizados

Campo	Valores	
Flgs	e	1
	e*	2
	ed	3
	es	4
	eg	5
	er	6
	M	7
	M*	8
	Md	9
	Mi	10
	Ms	11
Dur	0 – 0.1	0
	0.1 – 0.2	1
	0.2 – 0.3	2
	...	
	4.9 – 5.0	49
SrcAddr	192.168.1.66	3232235842
DstAddr	192.168.1.110	3232235886
	192.168.1.120	3232235896
Proto	TCP	1
Sport	53738	53738
	53741	53741
	56280	56280
	56283	56283
Dport	502	502
STos, DTos	0 – 25	0
	25 – 50	1
	...	
	250 – 275	10
sDSb, dDSb	0	0
	cs0	1
	cs6	2
	ef	3
sTtl, dTtl	64	64
	128	128
sHops, dHops	0 – 3	0
	3 – 6	1
	...	
	27 – 30	9

Tabla D.4: Valores de las variables utilizadas (1)

Campo	Valores	
sIpId, dIpId	0 – 5000	0
	5000 – 10000	1
	...	
	65000 – 70000	13
Cause	Start	1
	Status	2
	Stop	3
	Close	4
	Error	5
SrcPkts, DstPkts	0 – 5	0
	5 – 10	1
	...	
	40 – 45	8
SrcBytes, DstBytes	0 – 100	0
	200 – 300	1
	...	
	3000 – 3100	30
PCRatio	-1 – -0.9	0
	-0.9 – -0.8	1
	...	
	0.9 – 1	20
SrcLoad, DstLoad	0 – 100	0
	100 – 200	1
	...	
	18500 - 18600	185
SrcLoss, DstLoss	0 – 3	0
	3 – 6	1
	...	
	15 – 18	5
SrcRetra, DstRetra	0	0
	cs0	1
SrcGap, DstGap	0 – 50	0
	50 – 100	1
	...	
	450 - 500	9
SIntPkt, DIntPkt	0 – 100	0
	100 – 200	1
	...	
	1000 – 1100	10
SrcJitter, DstJitter	0 – 100	0
	100 – 200	1
	...	
	1000 – 1100	10

Tabla D.5: Valores de las variables utilizadas (2)

Campo	**Valores**	
State	CON	0
	FIN	1
	RST	2
SrcWin	0	0
	1868800	1
	3737600	2
DstWin	0	0
	2034	1
	2035	2
	2036	3
	2048	4

Tabla D.6: Valores de las variables utilizadas (3)